

DELPHION**Select CR****RESEARCH****PRODUCTS****INSIDE DELPHION****Log Out****Work Files****Saved Searches**

My Account

Search: Quick/Number Boolean Advanced Der

The Delphion Integrated View

Get Now: ☒ PDF | [More choices...](#)Tools: Add to Work File: [Create new Work](#)View: [INPADOC](#) | Jump to: [Top](#) Go to: [Derwent](#)[Email](#)

Title: JP2001043290A2: METHODS FOR PRESENTING, INSERTING, DELE
 UPDATING TABLE FORM DATA, TRANSACTION PROCESSING MET
 UTILIZING THE METHODS, PARALLEL PROCESSING METHOD AND
 STORAGE MEDIUM STORING PROGRAM FOR PROVIDING THE ME

Derwent Title: Table format data presenting inserting, deleting and updating method
[\[Derwent Record\]](#)

Country: JP Japan

Kind: A2 Document Laid open to Public inspection i

Inventor: KOSHO SHINJI;

Assignee: TAABO DATA LABORATORY KK
[News, Profiles, Stocks and More about this company](#)

Published / Filed: 2001-02-16 / 1999-07-29

Application Number: JP1999000215450

IPC Code: [G06F 19/00](#); [G06F 12/00](#); [G06F 17/30](#);

ECLA Code: G06F17/30S1R;

Priority Number: 1999-07-29 JP1999000215450

Abstract: PROBLEM TO BE SOLVED: To make suitable parallel processing or transaction processing by specifying the item value of an array by applying prescribed offset to a subscript corresponding to a subscript conversion array.

SOLUTION: First of all, a storage position number showing the position of the subscript is initialized (S1101) and next, while referring to the elements of start and end arrays, which of ranges determined by the values of start position and end position arrays includes the subscript is specified (S1102). Next, on the basis of that range, the correspondent offset value is extracted (S1103) and a value adding the offset value to the subscript is provided (S1104). Since this value with the added offset becomes a converted subscript, an element in a substance array is specified (S1105) by utilizing this value. Next, concerning the required subscript, '1' is added to the storage position number and by repeating similar processing (S1106 and S1107), the element in the substance array can be provided while maintaining a logical relation.

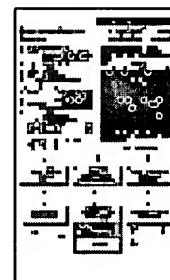
COPYRIGHT: (C)2001,JPO

INPADOC None **Get Now:** [Family Legal Status Report](#)

Legal Status:

Designated CA CN EP KR US

Country:

**BEST AVAILABLE COPY**

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開2001-43290

(P2001-43290A)

(43)公開日 平成13年2月16日(2001.2.16)

(51)Int.Cl. ⁷	識別記号	F I	テーマコード*(参考)
G 0 6 F 19/00		G 0 6 F 15/22	3 1 0 A 5 B 0 7 5
12/00	5 1 2	12/00	5 1 2 5 B 0 8 2
17/30		15/40	3 8 0 D
		15/401	3 4 0 C

審査請求 未請求 請求項の数29 O L (全 44 頁)

(21)出願番号 特願平11-215450

(22)出願日 平成11年7月29日(1999.7.29)

(71)出願人 599074648

ターボデータラボラトリー有限会社

東京都台東区松が谷1-9-12 S P K ビ

ルディング604号

(72)発明者 古庄 晋二

神奈川県横浜市神奈川区松見町4丁目1101

番地7 コートハウス菊名804号

(74)代理人 100103632

弁理士 龜田 英一郎 (外1名)

Fターム(参考) 5B075 PP02 PP03 PP12 PP13 PQ02

PQ15 PQ46 QT06

5B082 GA03 GA14 GA16 GA18

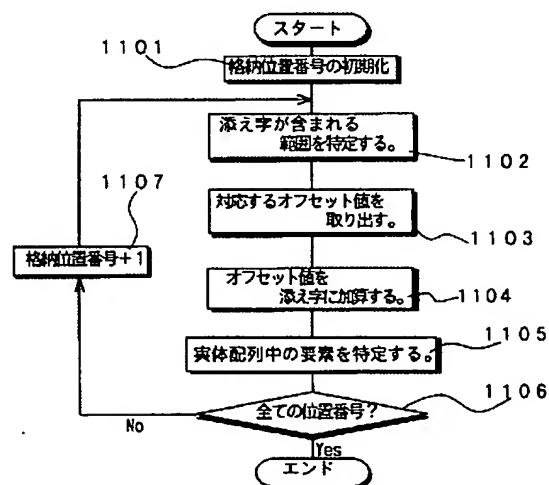
(54)【発明の名称】 表形式データの提示方法、挿入方法、削除方法、更新方法およびこれら方法を利用したトランザクション処理方法、並列処理方法、並びに、上記方法を実現するプログラムを記憶した記憶媒体

(57)【要約】

【課題】 表形式データにおいて、データの挿入、削除および更新の処理を高速かつ適切にす。

【解決手段】 CPU12は、レコード番号を添え字として受け入れ、当該添え字の範囲に対応するオフセット値を与えるための添え字変換配列を生成し、挿入される項目値の位置を示す挿入位置を特定し、添え字変換配列において、挿入位置に関して、対応する添え字の範囲を確定するとともに、配列の末尾を特定するためのオフセット値を与え、添え字変換配列において、挿入位置に対応するレコード番号より大きな番号を有するものに関して、対応する添え字の範囲をインクリメントするとともに、受け入れられた添え字をデクリメントするためのオフセット値を与え、上記末尾の位置に、挿入すべき項目値を配置し、添え字に、添え字変換配列中の添え字の範囲にしたがったオフセット値を与える。これにより、オフセット値が与えられた添え字を用いて配列中の項目値が特定される。

図 11 (a)



【特許請求の範囲】

【請求項1】 項目とこれに含まれる項目値とを含むレコードの配列として表わされる表形式データの任意の位置に、項目値を挿入する表形式データの挿入方法であって、

レコード番号を添え字として受け入れ、当該添え字の範囲に対応するオフセット値を与えるように構成された添え字変換配列を生成し、

挿入される項目値の位置を示す挿入位置を特定し、

前記添え字変換配列において、前記挿入位置に関して、対応する添え字の範囲を確定するとともに、前記配列の末尾以降の所定の位置を特定するためのオフセット値を与え、

前記添え字変換配列において、前記挿入位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を上方にシフトするとともに、受け入れられた添え字をデクリメントするためのオフセット値を与え、

前記配列の末尾以降の所定の位置に、前記挿入すべき項目値を配置し、

前記添え字に、添え字変換配列中の添え字の範囲にしたがったオフセット値が与えられ、当該オフセット値が与えられた添え字により配列中の項目値が特定されることを特徴とする表形式データの挿入方法。

【請求項2】 項目とこれに含まれる項目値とを含むレコードの配列として表わされる表形式データにおいて、任意の位置の項目値を削除する表形式データの削除方法であって、

レコード番号を添え字として受け入れ、受け入れられた添え字の範囲に対応するオフセット値を与えるように構成された添え字変換配列を生成し、

削除すべき項目値の位置を示す削除位置を特定し、

前記添え字変換配列において、削除位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を下方にシフトするとともに、受け入れられた添え字をインクリメントするためのオフセット値を与え、

前記添え字に、添え字変換配列中の添え字の範囲にしたがったオフセット値が与えられ、当該オフセット値が与えられた添え字により配列中の項目値が特定されることを特徴とする表形式データの削除方法。

【請求項3】 項目とこれに含まれる項目値とを含むレコードの配列として表わされる表形式データにおいて、任意の位置の項目値を更新する表形式データの更新方法であって、

レコード番号を添え字として受け入れ、受け入れられた添え字の範囲に対応するオフセット値を与えるように構成された添え字変換配列を生成し、（1）更新すべき項目値の位置を削除位置として仮定して、

前記添え字変換配列において、削除位置に対応するレコ

ード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を下方にシフトするとともに、受け入れられた添え字をインクリメントするためのオフセット値を与え、（2）更新すべき項目値の位置を挿入位置として仮定して、

前記添え字変換配列において、前記挿入位置に関して、対応する添え字の範囲を確定するとともに、前記配列の末尾以降の所定の位置を特定するためのオフセット値を与え、

10 前記添え字変換配列において、前記挿入位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を上方にシフトするとともに、受け入れられた添え字をデクリメントするためのオフセット値を与え、

かつ、前記配列の末尾以降の所定の位置に、前記更新すべき項目値を配置し、前記（1）および（2）、或いは、（2）および（1）を順次実行して、

前記添え字に、添え字変換配列中の添え字の範囲にしたがったオフセット値が与えられ、当該オフセット値が与えられた添え字により配列中の項目値が特定されることを特徴とする表形式データの更新方法。

20 【請求項4】 項目とこれに含まれる項目値とを含むレコードの配列として表わされる表形式データの項目値の値を変換する表形式データの値変換方法であって、前記項目値の範囲に対応するオフセット値を与えるように構成された値変換配列を生成し、

レコード番号を添え字として受け入れ、前記配列中の、当該添え字に対応する項目値に、前記項目値の範囲に対応したオフセット値を与えるように構成されたことを特徴とする表形式データの値変換方法。

30 【請求項5】 項目とこれに含まれる項目値とを含むレコードの配列として表わされる表形式データにおいて、任意のレコード番号に対応する項目値を特定するように構成された表形式データのデータ構造であって、当該表形式データを、特定の項目に属する項目値に対応した項目値番号の順に当該項目値が格納されている第1の実体配列を含む値リストと、一意的なレコード番号の順に、当該項目値番号を指示するためのポインタ値が格納された第2の実体配列を含むポインタ配列とからなる一以上の情報ブロックに分割するように構成し、

40 前記ポインタ配列に、入力されたレコード番号を添え字として、当該添え字の範囲に対応する第1のオフセットを与えるように構成された第1の添え字変換配列を形成するとともに、当該添え字変換配列を経てオフセットが与えられた添え字により特定される、第2の実体配列中のポインタ値を値として、当該値の範囲に対応する第2のオフセットを与えるように構成された値変換配列を形成し、

50 前記値リストに、入力された、ポインタ配列の出力を添え字として、当該添え字の範囲に対応する第3のオフセ

ットを与えるように構成された第2の添え字変換配列を形成し、当該第2の添え字変換配列を経てオフセットが与えられたポインタ配列の出力により、第1の実体配列中の項目値が特定されるように構成されたことを特徴とする表形式データのデータ構造。

【請求項6】 前記添え字変換配列の各々が、所定の範囲にそれぞれ含まれる添え字の最小値を示す開始位置からなる開始位置配列、および／または、当該所定の範囲に含まれる添え字の最大値を示す終了位置からなる終了位置配列と、対応するオフセット値からなるオフセット配列とから構成されることを特徴とする請求項5に記載のデータ構造。

【請求項7】 前記値変換配列が、所定の範囲にそれぞれ含まれる値の最小値を示す開始位置からなる開始位置配列、および／または、当該所定の範囲に含まれる値の最大値を示す終了位置からなる終了位置配列と、対応するオフセット値からなるオフセット配列とから構成されることを特徴とする請求項5に記載のデータ構造。

【請求項8】 請求項5ないし7の何れか一項に記載のデータ構造にて表わされる表形式データの任意の位置に、項目値を挿入する表形式データの挿入方法であって、(1)前記値リストに関して、

挿入される項目値の位置を示す挿入位置を特定し、前記第2の添え字変換配列において、前記挿入位置に関して、対応する添え字を確定するとともに、前記第1の実体配列の末尾以降の所定の位置を特定するための第3のオフセット値を与え、

前記第2の添え字変換配列において、前記挿入位置に対応する添え字の値より大きなものに関して、対応する添え字の範囲を、当該範囲を確定する値が大きくなるように上方にシフトするとともに、受け入れられた添え字をデクリメントするための第3のオフセット値を与え、前記第1の実体配列の末尾以降の所定の位置に、前記挿入すべき項目値を配置し、(2)前記ポインタ配列に関して、

挿入されるレコード番号に対応するポインタ値の位置を示す挿入位置を特定し、

前記第1の添え字変換配列において、前記挿入位置に関して、対応する添え字を確定するとともに、前記第1の実体配列の末尾以降の所定の位置を特定するための第1のオフセット値を与え、

前記第1の添え字変換配列において、前記挿入位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を、当該範囲を確定する値が大きくなるように上方にシフトするとともに、受け入れられた添え字をデクリメントするための第1のオフセット値を与え、

前記第2の実体配列の末尾以降の所定の位置に、現存するポインタ値より大きな新たなポインタ値を配置し、かつ、(3)前記ポインタ配列に関して、

前記値変換配列に、前記値リストにおける挿入位置に対応するポインタ値以上を有するものに対して、これをインクリメントする第2のオフセット値を与え、

前記値変換配列に、前記新たなポインタ値が、前記挿入位置に対応する位置を特定するような第2のオフセット値を与えることを特徴とする表形式データのデータ挿入方法。

【請求項9】 m個のレコード番号と、n個の項目値とを備え、

10 前記値リスト中の項目値の挿入位置が、 i ($0 \leq i \leq n-1$)であり、第1の実体配列の末尾nに項目値が配置され、かつ、レコード番号の挿入位置が j ($0 \leq j \leq m-1$)であり、第2の実体配列の末尾mにポインタ値が配置される場合に、(1)前記値リストの第2の添え字変換配列において、

値の範囲が $(i-1)$ 以下の場合に、第3のオフセット値として0を与え、

値が i である場合に、第3のオフセット値として $(n-i)$ を与え、かつ、

20 値の範囲が $(i+1)$ 以上n以下である場合に、第3のオフセット値として (-1) を与え、(2)前記ポインタ配列の第1の添え字変換配列において、

添え字の範囲が $(j-1)$ 以下の場合に、第1のオフセット値として0を与え、

添え字が j である場合に、第1のオフセット値として $(m-j)$ を与え、かつ、

添え字が $(j+1)$ 以上m以下の場合に、第1のオフセット値として (-1) を与え、(3)前記ポインタ配列の値変換配列において、

30 値の範囲が $(i-1)$ の場合に、第2のオフセット値として0を与え、

値の範囲が i 以上 $(n-1)$ 以下である場合に、第2のオフセット値として1を与え、かつ、

値がnである場合に、第2のオフセット値として $(i-n)$ を与えることを特徴とする請求項8に記載の表形式データの挿入方法。

【請求項10】 m個のレコード番号と、n個の項目値とを備え、

40 前記値リスト中の項目値の挿入位置が、 i ($0 \leq i \leq n-1$)であり、第1の実体配列の末尾以降の所定の位置 z ($z \geq n$)に項目値が配置され、かつ、レコード番号の挿入位置が j ($0 \leq j \leq m-1$)であり、第2の実体配列の末尾以降の所定の位置 x ($x \geq m$)にポインタ値が配置される場合に、(1)前記値リストの第2の添え字変換配列において、

値の範囲が $(i-1)$ 以下の場合に、第3のオフセット値として0を与え、

値が i である場合に、第3のオフセット値として $(z-i)$ を与え、かつ、

50 値の範囲が $(i+1)$ 以上n以下である場合に、第3の

オフセット値として (-1) を与え、(2)前記ポインタ配列の第1の添え字変換配列において、添え字の範囲が $(j-1)$ 以下の場合に、第1のオフセット値として0を与え、
 添え字が j である場合に、第1のオフセット値として $(x-j)$ を与え、かつ、
 添え字が $(j+1)$ 以上 m 以下の場合に、第1のオフセット値として (-1) を与えるとともに、(3)前記ポインタ配列の値変換配列において、
 値の範囲が $(i-1)$ の場合に、第2のオフセット値として0を与え、
 値の範囲が i 以上 $(n-1)$ 以下である場合に、第2のオフセット値として1を与え、かつ、
 値が y (ただし、 y は、第2の実体配列の位置 x に格納されたポインタ値)である場合に、第2のオフセット値として $(i-y)$ を与えることを特徴とする請求項8に記載の表形式データの挿入方法。

【請求項11】 請求項5ないし7の何れか一項に記載のデータ構造にて表わされる表形式データの任意の位置の項目値を削除する表形式データの削除方法であって、
 前記ポインタ配列に関して、
 削除されるポインタ値の位置を示す削除位置を特定し、
 前記第1の添え字変換配列において、前記削除に関して、対応する添え字を確定し、
 前記第1の添え字変換配列において、前記削除位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を、当該範囲を確定する値が小さくなるように下方にシフトするとともに、受け入れられた添え字をインクリメントするための第1のオフセット値を与えることを特徴とする表形式データの削除方法。

【請求項12】 m 個のレコード番号と、 n 個の項目値とを備え、
 レコード番号の削除位置が j ($0 \leq j \leq m-1$)である場合に、
 前記ポインタ配列の第1の添え字変換配列において、添え字の範囲が $(j-1)$ 以下の場合に、第1のオフセット値として0を与え、かつ、
 添え字の範囲が j 以上 $(m-2)$ 以下である場合に、オフセット値として1を与えることを特徴とする請求項11に記載の表形式データの削除方法。

【請求項13】 請求項5ないし7の何れか一項に記載のデータ構造にて表わされる表形式データの任意の位置の項目値を更新する表形式データの更新方法であって、
 (A)(1)前記値リストに関して、
 更新される項目値を挿入される項目値と仮定し、当該項目値の挿入すべき位置を示す挿入位置を特定し、
 前記第2の添え字変換配列において、前記挿入位置に関して、対応する添え字を確定するとともに、前記第1の実体配列の末尾以降の所定の位置を特定するためのオフ

セット値を与え、
 前記第2の添え字変換配列において、前記挿入位置に対応する添え字の値より大きなものに関して、対応する添え字の範囲を、当該範囲を確定する値が大きくなるように上方にシフトするとともに、受け入れられた添え字をデクリメントするための第3のオフセット値を与え、
 前記第1の実体配列の末尾以降の所定の位置に、前記挿入すべき項目値を配置し、(2)前記ポインタ配列に関して、
 更新されるポインタ値を挿入されるポインタ値と仮定し、当該ポインタ値の挿入すべき位置を示す挿入位置を特定し、
 前記第1の添え字変換配列において、前記挿入位置に関して、対応する添え字を確定するとともに、前記第1の実体配列の末尾以降の所定の位置を特定するための第1のオフセット値を与え、
 前記第1の添え字変換配列において、前記挿入位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を、当該範囲を確定する値が大きくなるように上方にシフトするとともに、受け入れられた添え字をデクリメントするための第1のオフセット値を与え、
 前記第2の実体配列の末尾以降の所定の位置に、現存するポインタ値より大きな新たなポインタ値を配置し、かつ、(3)前記ポインタ配列に関して、
 前記値変換配列に、前記値リストにおける挿入位置に対応するポインタ値以上を有するものに対して、これをインクリメントする第2のオフセット値を与え、
 前記値変換配列に、前記新たなポインタ値が、前記挿入位置に対応する位置を特定するような第2のオフセット値を与え、(B)前記ポインタ配列に関して、
 更新すべきポインタ値の位置を、削除位置として、前記挿入位置を考慮して特定し、
 前記第1の添え字変換配列において、前記削除位置に関して、対応する添え字を確定し、
 前記第1の添え字変換配列において、前記削除位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を、当該範囲を確定する値が小さくなるように下方にシフトするとともに、受け入れられた添え字をインクリメントするための第1のオフセット値を与え、前記(A)および(B)、或いは、前記(B)および(A)の何れかを順次実行するように構成されたことを特徴とする表形式データの更新方法。
 【請求項14】 m 個のレコード番号と、 n 個の項目値とを備え、
 前記値リスト中の項目値の更新位置が、 i ($0 \leq i \leq n-1$)であり、第1の実体配列の末尾 n に項目値が配置され、かつ、レコード番号の更新位置が j ($0 \leq j \leq m-1$)であり、第2の実体配列の末尾 m にポインタ値が配置される場合に、(1)前記値リストの第2の添え字

変換配列において、
 値の範囲が $(i-1)$ 以下の場合に、第3のオフセット値として0を与え、
 値が i である場合に、第3のオフセット値として $(n-i)$ を与え、かつ、
 値の範囲が $(i+1)$ 以上 n 以下である場合に、第3のオフセット値として (-1) を与え、(2) 前記ポインタ配列の第1の添え字変換配列において、
 添え字の範囲が $(j-1)$ 以下の場合に、第1のオフセット値として0を与え、
 添え字が j である場合に、第1のオフセット値として $(m-j)$ を与え、かつ、
 添え字が $(j+1)$ 以上 $(m+1)$ 以下の場合に、第1のオフセット値として0を与え、(3) 前記ポインタ配列の値変換配列において、
 値の範囲が $(i-1)$ の場合に、第2のオフセット値として0を与え、
 値の範囲が i 以上 $(n-1)$ 以下である場合に、第2のオフセット値として1を与え、かつ、
 値が n である場合に、第2のオフセット値として $(i-n)$ を与えることを特徴とする請求項13に記載の表形式データの更新方法。
 【請求項15】 m 個のレコード番号と、 n 個の項目値とを備え、
 前記値リスト中の項目値の更新位置が、 i ($0 \leq i \leq n-1$) であり、第1の実体配列の末尾以降の所定の位置 z ($z \geq n$) に更新された項目値が配置され、かつ、レコード番号の更新位置が j ($0 \leq j \leq m-1$) であり、
 第2の実体配列の末尾以降の所定の位置 x ($x \geq m$) に更新されたポインタ値が配置される場合に、(1) 前記値リストの第2の添え字変換配列において、
 値の範囲が $(i-1)$ 以下の場合に、第3のオフセット値として0を与え、
 値が i である場合に、第3のオフセット値として $(z-i)$ を与え、かつ、
 値の範囲が $(i+1)$ 以上 n 以下である場合に、第3のオフセット値として (-1) を与え、(2) 前記ポインタ配列の第1の添え字変換配列において、
 添え字の範囲が $(j-1)$ 以下の場合に、第1のオフセット値として0を与え、
 添え字が j である場合に、第1のオフセット値として $(x-j)$ を与え、かつ、
 添え字が $(j+1)$ 以上 $(m+1)$ 以下の場合に、第1のオフセット値として0を与えるとともに、(3) 前記ポインタ配列の値変換配列において、
 値の範囲が $(i-1)$ の場合に、第2のオフセット値として0を与え、
 値の範囲が i 以上 $(n-1)$ 以下である場合に、第2のオフセット値として1を与え、かつ、
 値が y (ただし、 y は、第2の実体配列の位置 x に格納

されたポインタ値)である場合に、第2のオフセット値として $(i-y)$ を与えることを特徴とする請求項13に記載の表形式データの挿入方法。

【請求項16】 請求項5ないし7の何れか一項に記載のデータ構造にて表わされる表形式データの任意の位置の項目値を削除する表形式データの削除方法であって、
 前記レコード番号に関して、レコード番号自体を配置した第4の実体配列と、当該レコード番号を特定するための添え字の範囲にしたがって、所定のオフセット値を与える第3の添え字変換配列とを設け、
 前記削除すべきレコード番号の位置を示す削除位置を特定し、
 前記第3の添え字変換配列において、前記削除に関して対応する添え字を確定し、
 前記第3の添え字変換配列において、前記削除位置に対応する添え字より大きな値を有するものに関して、対応する添え字の範囲を、当該範囲を確定する値が小さくなるように下方にシフトするとともに、受け入れられた添え字をインクリメントするための第4のオフセット値を与えることを特徴とする表形式データの削除方法。

【請求項17】 項目とこれに含まれる項目値とを含むレコードの配列として表わされる表形式データの任意の位置の項目値を削除する表形式データの削除方法であって、
 前記レコード番号に関して、レコード番号自体を配置した配列と、当該レコード番号を特定するための添え字の範囲にしたがって、所定のオフセット値を与える添え字変換配列とを設け、
 前記削除すべきレコード番号の位置を示す削除位置を特定し、
 前記添え字変換配列において、前記削除に関して対応する添え字を確定し、
 前記添え字変換配列において、前記削除位置に対応する添え字より大きな値を有するものに関して、対応する添え字の範囲を、当該範囲を確定する値が小さくなるように下方にシフトするとともに、受け入れられた添え字をインクリメントするためのオフセット値を与えることを特徴とする表形式データの削除方法。

【請求項18】 請求項8ないし15の何れか一項に記載の方法により表形式データにおけるデータの挿入、削除および/または更新をなし、ロールバックおよびコミットの何れかの場合には、前記添え字変換配列および値変換配列を廃棄することを特徴とするトランザクション処理方法。

【請求項19】 請求項8ないし15の何れか一項に記載の方法により表形式データにおけるデータの挿入、削除および/または更新をなし、ロールバックの場合に、前記添え字変換配列および値変換配列を廃棄することを特徴とするトランザクション方法。

【請求項20】 請求項5ないし7のデータ構造を有す

る表形式データに関して、複数のプロセスにより同時に処理が実行されるプロセスの並列処理方法であって、前記プロセスのうち、データの挿入、削除および/または更新を伴うものは、前記ポインタ配列において、第1の添え字変換配列、第2の実体配列および値変換配列を経るよう構成されるとともに、前記値リストにおいて、第2の添え字変換配列および第1の実体配列を経るよう構成され、

その一方、前記プロセスのうち、データの挿入、削除および/または更新を伴わないものは、前記ポインタ配列において、第2の実体配列を経るとともに、前記値リストにおいて、前記値リストにおいて、第1の実体配列を経るよう構成されたことを特徴とするプロセスの並列処理方法。

【請求項21】 前記データの挿入、削除および/または更新を伴うプロセスが、請求項8ないし請求項15の何れか一項に記載された方法を利用することを特徴とする請求項20に記載のプロセス並列処理方法。

【請求項22】 請求項5ないし7のデータ構造を有する表形式データに関して、レコードのロックをなす方法であって、

ロックを管理する配列を含む情報ブロックを設け、前記情報ブロックの配列中に、前記レコード番号の各々に対応して、ロックの種別を示す項目値を配置することを特徴とする方法。

【請求項23】 項目とこれに含まれる項目値とを含むレコードの配列として表わされる表形式データの任意の位置に、項目値を挿入する表形式データのデータ挿入プログラムを記憶した、コンピュータにより読み出し可能な記憶媒体であって、

レコード番号を添え字として受け入れ、当該添え字の範囲に対応するオフセット値を与えるよう構成された添え字変換配列を生成し、

挿入される項目値の位置を示す挿入位置を特定し、前記添え字変換配列において、前記挿入位置に関して、対応する添え字の範囲を確定するとともに、前記配列の末尾を特定するためのオフセット値を与え、

前記添え字変換配列において、前記挿入位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を上方にシフトするとともに、受け入れられた添え字をデクリメントするためのオフセット値を与え、

前記配列の末尾に、前記挿入すべき項目値を配置し、前記添え字に、添え字変換配列中の添え字の範囲にしたがったオフセット値が与えられ、当該オフセット値が与えられた添え字により配列中の項目値が特定されることを特徴とする表形式データのデータ挿入プログラムを記憶した記憶媒体。

【請求項24】 項目とこれに含まれる項目値とを含むレコードの配列として表わされる表形式データにお

て、任意の位置の項目値を削除する表形式データのデータ削除プログラムを記憶した、コンピュータにより読み出し可能な記憶媒体であって、

レコード番号を添え字として受け入れ、受け入れられた添え字の範囲に対応するオフセット値を与えるよう構成された添え字変換配列を生成し、

削除すべき項目値の位置を示す削除位置を特定し、

前記添え字変換配列において、削除位置に対応するレコード番号より大きなレコード番号を有するものに関し

て、対応する添え字の範囲を下方にシフトするとともに、受け入れられた添え字をインクリメントするためのオフセット値を与え、

前記添え字に、添え字変換配列中の添え字の範囲にしたがったオフセット値が与えられ、当該オフセット値が与えられた添え字により配列中の項目値が特定されることを特徴とする表形式データのデータ削除プログラムを記憶した記憶媒体。

【請求項25】 項目とこれに含まれる項目値とを含むレコードの配列として表わされる表形式データにおいて、任意の位置の項目値を更新する表形式データのデータ更新プログラムを記憶した、コンピュータにより読み出し可能な記憶媒体であって、

レコード番号を添え字として受け入れ、受け入れられた添え字の範囲に対応するオフセット値を与えるよう構成された添え字変換配列を生成し、(1)更新すべき項目値の位置を削除位置として仮定して、

前記添え字変換配列において、削除位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を下方にシフトするとともに、受け入れられた添え字をインクリメントするための

オフセット値を与え、(2)更新すべき項目値の位置を挿入位置として仮定して、

前記添え字変換配列において、前記挿入位置に関して、対応する添え字の範囲を確定するとともに、前記配列の末尾を特定するためのオフセット値を与え、

前記添え字変換配列において、前記挿入位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を上方にシフトするとともに、受け入れられた添え字をデクリメントするためのオフセット値を与え、

かつ、前記配列の末尾に、前記更新すべき項目値を配置し、

前記(1)および(2)、或いは、(2)および(1)を順次実行して、

前記添え字に、添え字変換配列中の添え字の範囲にしたがったオフセット値が与えられ、当該オフセット値が与えられた添え字により配列中の項目値が特定されることを特徴とする表形式データのデータ更新プログラムを記憶した記憶媒体。

【請求項26】 項目とこれに含まれる項目値とを含む

レコードの配列として表わされる表形式データの項目値の値を変換する表形式データの値変換プログラムを記憶した、コンピュータにより読み出し可能な記憶媒体であって、

項目値の範囲に対応するオフセット値を与えるように構成された値変換配列を生成し、

レコード番号を添え字として受け入れ、前記配列中の、当該添え字に対応する項目値に、前記項目値の範囲に対応したオフセット値を与えるように構成されたことを特徴とする表形式データの値変換プログラムを記憶した記憶媒体。

【請求項27】 請求項5ないし7の何れか一項に記載のデータ構造にて表わされる表形式データの任意の位置に、項目値を挿入する表形式データのデータ挿入プログラムを記憶したコンピュータにより読み出し可能な記憶媒体であって、

(1) 前記値リストに関して、挿入される項目値の位置を示す挿入位置を特定し、前記第2の添え字変換配列において、前記挿入位置に関して、対応する添え字を確定するとともに、前記第1の実体配列の末尾以降の所定の位置を特定するための第3のオフセット値を与え、

前記第2の添え字変換配列において、前記挿入位置に対応する添え字の値より大きなものに関して、対応する添え字の範囲を、当該範囲を確定する値が大きくなるように上方にシフトするとともに、受け入れられた添え字をデクリメントするための第3のオフセット値を与え、

前記第1の実体配列の末尾以降の所定の位置に、前記挿入すべき項目値を配置し、(2) 前記ポインタ配列に関して、

挿入されるレコード番号に対応するポインタ値の位置を示す挿入位置を特定し、

前記第1の添え字変換配列において、前記挿入位置に関して、対応する添え字を確定するとともに、前記第1の実体配列の末尾以降の所定の位置を特定するための第1のオフセット値を与え、

前記第1の添え字変換配列において、前記挿入位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を、当該範囲を確定する値が大きくなるように上方にシフトするとともに、受け入れられた添え字をデクリメントするための第1のオフセット値を与え、

前記第2の実体配列の末尾以降の所定の位置に、現存するポインタ値より大きな新たなポインタ値を配置し、かつ、(3) 前記ポインタ配列に関して、

前記値変換配列に、前記値リストにおける挿入位置に対応するポインタ値以上を有するものに対して、これをインクリメントする第2のオフセット値を与え、

前記値変換配列に、前記新たなポインタ値が、前記挿入位置に対応する位置を特定するような第2のオフセット

値を与えることを特徴とする表形式データのデータ挿入プログラムを記憶した記憶媒体。

【請求項28】 請求項5ないし7の何れか一項に記載のデータ構造にて表わされる表形式データの任意の位置の項目値を削除する表形式データのデータ削除プログラムを記憶した、コンピュータにより読み出し可能な記憶媒体であって、

前記ポインタ配列に関して、削除されるポインタ値の位置を示す削除位置を特定し、前記第1の添え字変換配列において、前記削除に関して、対応する添え字を確定し、

前記第1の添え字変換配列において、前記削除位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を、当該範囲を確定する値が小さくなるように下方にシフトするとともに、受け入れられた添え字をインクリメントするための第1のオフセット値を与えることを特徴とする表形式データのデータ削除プログラムを記憶した記憶媒体。

【請求項29】 請求項5ないし7の何れか一項に記載のデータ構造にて表わされる表形式データの任意の位置の項目値を更新する表形式データのデータ更新プログラムを記憶した、コンピュータにより読み出し可能な記憶媒体であって、(A) (1) 前記値リストに関して、更新される項目値を挿入される項目値と仮定し、当該項目値の挿入すべき位置を示す挿入位置を特定し、前記第2の添え字変換配列において、前記挿入位置に関して、対応する添え字を確定するとともに、前記第1の実体配列の末尾以降の所定の位置を特定するためのオフセット値を与え、

前記第2の添え字変換配列において、前記挿入位置に対応する添え字の値より大きなものに関して、対応する添え字の範囲を、当該範囲を確定する値が大きくなるように上方にシフトするとともに、受け入れられた添え字をデクリメントするための第3のオフセット値を与え、

前記第1の実体配列の末尾以降の所定の位置に、前記挿入すべき項目値を配置し、(2) 前記ポインタ配列に関して、

更新されるポインタ値を挿入されるポインタ値と仮定し、当該ポインタ値の挿入すべき位置を示す挿入位置を特定し、

前記第1の添え字変換配列において、前記挿入位置に関して、対応する添え字を確定するとともに、前記第1の実体配列の末尾以降の所定の位置を特定するための第1のオフセット値を与え、

前記第1の添え字変換配列において、前記挿入位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を、当該範囲を確定する値が大きくなるように上方にシフトするとともに、受け入れられた添え字をデクリメントするための第1のオフセット値を与え、

前記第2の実体配列の末尾以降の所定の位置に、現存するポインタ値より大きな新たなポインタ値を配置し、かつ、(3)前記ポインタ配列に関して、前記値変換配列に、前記値リストにおける挿入位置に対応するポインタ値以上を有するものに対して、これをインクリメントする第2のオフセット値を与え、前記値変換配列に、前記新たなポインタ値が、前記挿入位置に対応する位置を特定するような第2のオフセット値を与え、(B)前記ポインタ配列に関して、更新すべきポインタ値の位置を、削除位置として、前記挿入位置を考慮して特定し、前記第1の添え字変換配列において、前記削除位置に関して、対応する添え字を確定し、前記第1の添え字変換配列において、前記削除位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を、当該範囲を確定する値が小さくなるように下方にシフトするとともに、受け入れられた添え字をインクリメントするための第1のオフセット値を与え、前記(A)および(B)、或いは、前記(B)および(A)の何れかを順次実行するように構成されたことを特徴とする表形式データのデータ更新プログラムを記憶した記憶媒体。

【発明の詳細な説明】

【0001】

【産業上の技術分野】本発明は、コンピュータのような情報処理装置を用いて大量のデータを処理するデータ処理方法及びデータ処理装置に関し、より詳細には、データベースを構成する表形式データの更新、削除、挿入およびトランザクション処理に関する。

【0002】

【従来の技術】データベースは種々の用途に用いられているが、中規模ないし大規模システムにおいては、論理的な矛盾が排除できるリレーショナルデータベース(RDB)の使用が主流となっている。たとえば、RDBは飛行機の座席予約等のシステムに利用されている。この場合、キー項目を指定することにより、(多くの場合1件の)ターゲットを迅速に検索することもでき、或いは、予約の確定、キャンセル或いは変更などを行うことができる。しかし、近年、データベースを、上記の如く単純に個々のレコードデータを管理する用途(基幹系業務)以外に、分析(情報系業務)にも使用する傾向が顕著である。例えば、上記データベースの所有者は、時間帯、路線、季節などによって搭乗率がどう変化するかを、クロス集計を作成して分析することを試みるであろう。

【0003】

【発明が解決しようとする課題】ところが、従来のRDBは、ローカルメモリを使用しコミットすることによりその内容を確定しているため、トランザクション処理を主体とする基幹業務には適しても、その構造上、大規模

データの分析(情報系)処理には向かない。たとえば、任意の項目でソートを行おうとすれば、理論上、(たとえばn件のレコードでは)、 $n \times \log(n)$ の時間がかかり、大規模なデータほど処理効率が低下する。(現実の製品では、理論よりもさらに急速に処理性能の低下が進む。)そして、この制約($=n \times \log(n)$ の時間)はソートに限らず、部分集合の抽出(=検索)など一般的な操作全般に出現する。

【0004】もちろん、この制約を回避すべく、さまざまな高速化方法が考案されてはいる。その好例としてビットマップインデックスがある。ビットマップインデックスは、各レコードが条件を満たすか否かを1ビットであらわしたビットマップであり、例えば「性別」という項目があった場合、「男性」のビットマップと、「女性」のビットマップが作成される。このとき、「女性」のレコードのみを取り出そうとすれば(つまり、検索すれば)、高速にその部分集合(すなわち、女性のレコードの集合)を得ることができる。しかし、各値毎にビットマップを作成するということはいつでも可能なわけではない。例えば、10億レコードで10億通りの値をとる項目(ID番号など)の場合、10億ビット(約125MB)のビットマップを10億個作成しなければならなくなる。つまり、ビットマップインデックスは非常に少数のバリエーションしか持たない項目(たとえば性別)でしか実現できないことが分かる。また、この方法は部分集合に対しては即座には適用できないことが明らかであるし、レコードの挿入・削除が頻発する場合にも使用できない。他の高速化手法も同様に何らかの制約を持っている。このような各種高速化技術の適用範囲の制約のため、ユーザーには全く同等に見える操作が、瞬時に終了する場合と何時間もかかる場合があり、処理時間の極端な非連続性(=予測困難さ)という深刻な問題が発生している。たとえば、現在商業的に入手可能なRDBで全体集合が1000万レコードを有するテーブルで測定したところ、全件(=1000万件)をヒットさせた場合、1秒しかかからなかったのに、わずか737件をヒットさせただけで61秒を要したという報告がある。では、900万件ヒットする場合、どのくらいの処理時間がかかるのであろう。言うまでもないことだが、検索を実行する前に予めヒットする件数を知ることにはできないから、ヒット件数と検索の所要時間の対応関数が分かっていたとしても、所要時間を確実に予測することはできない。そもそも、男性という条件を満たす500万件のレコードを抽出する時間(検索時間)と、女性という条件を満たす500万件のレコードを抽出する時間(検索時間)が同じであるという理論的根拠すらも希薄ではある。

【0005】このようなパフォーマンスの不安定さは、特にリアルタイム性を要求される制御システムにおいては致命的であり、例えば複数の発電所と非常に多数の電

力需要を電力線ネットワークで接続した大規模かつ複雑な電力供給制御システム、また例えば大規模な化学プラントの制御システムなど、多量のデータを管理・使用するシステムでありながらデータベースを使用できない例は数多いのである。もちろん、パフォーマンスが安定していても遅すぎれば使い物にならないことは明らかであるが、ここで強調したいのは、特殊な限定条件下のみでしか適用できないさまざまな高速化方法が導入された結果、データベースシステムはひどく複雑になったばかりか、前述の処理時間の極端な非連続性という重大な問題を抱え込んだということである。

【0006】さて、近年、前述のデータベースの分析的使用方法が一般化していることは前述のようであり、同時にデータの巨大化が急速に進行していることも周知の事実である。このような状況のなかで、RDBにおける前述の問題を克服するために、DWH（データウェアハウス）および多次元DB（MDB：Multi Dimensional Database）が出現した。DWHは、トランザクション処理機能などを外し、検索・集計性能に特化したチューニングを行ったRDBであると考えて差し支えない。したがって、 $n \times \log(n)$ の処理時間という根本的制約は残存したままであり、パフォーマンスは改善されたものの十分とは言えず、その非連続性の問題も解消されていない。さらに、基幹系の既存RDBと情報系のDWHという不可避な組み合わせにより、設備投資が大幅に増加し、情報管理の複雑化と運用コスト増加という新たな問題が発生している。

【0007】一方、MDBは指定された項目（次元とよばれている）の組み合わせによる基礎クロス集計を予め作成しておき、ユーザーの要求するクロス集計を基礎クロス集計から合成するものであると考えて良い。MDBは、クロス集計こそは満足なレスポンスで合成できるが、問題も多い。例えば、（集計から元データに戻することは不可能なので、特別な工夫をしない限り）元々のレコードに戻ることができない、（予めクロス集計を作成しておかねばならないため、）株価などの時時刻々と更新されるデータに対応できない、あるいは、予め指定した以外の項目（次元）のクロス集計は作成できないことなどの問題が挙げられる。もちろん、設備投資の増加、情報管理の複雑化と運用コストの上昇というDWHと同じ問題も発生している。現実を見ると、DWHやMDBは、前述のような制御システム、あるいは科学技術計算、金融工学など、コストを度外視しても性能を重視する分野においてすら稀にしか導入されない。このことは、これらの機能及び性能が前述の分野において実用に耐えないことの証左である。

【0008】このような問題を解決すべく、本発明者はすでに従来技術よりも略100～1000倍高速に表形式データを検索・集計・ソート・ジョインする方法（線形フィルター法）を考案し、これに関する特許出願をな

している（（特願平10-227278号）。線形フィルター法は、処理性能の高さと安定性、処理の統一性、データのコンパクトさと分割性、 $O(n)$ のアーキテクチャーであること、ソートの直列接続性、パイプライン処理や並行処理に適することなど非常に好ましい性質を多々備えている。そこで、本発明は、特に、上記線形フィルター法を用いて、データの挿入、削除および更新の処理を高速かつ適切になすことができる表形式データにおけるデータ挿入、削除および更新方法を提供することを目的とする。

【0009】また、広く一般に理解されているとおり、更新処理の実運用においては、複数のプロセスによる同一データへの同時アクセスを制御する排他処理、複数の処理を1つの有意義な処理ブロックとして一括してコミット（またはロールバック）するトランザクション処理を適切になすことが要求される。そこで、本発明は、上記トランザクション処理を適切になすことができる表形式データの挿入、削除および更新方法を提供することを目的とする。

【0010】

【課題を解決するための手段】本発明の目的は、項目とこれに含まれる項目値とを含むレコードの配列として表わされる表形式データの任意の位置に、項目値を挿入する表形式データの挿入方法であって、レコード番号を添え字として受け入れ、当該添え字の範囲に対応するオフセット値を与えるように構成された添え字変換配列を生成し、挿入される項目値の位置を示す挿入位置を特定し、前記添え字変換配列において、前記挿入位置に関して、対応する添え字の範囲を確定するとともに、前記配列の末尾以降の所定の位置を特定するためのオフセット値を与え、前記添え字変換配列において、前記挿入位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を上方にシフトするとともに、受け入れられた添え字をデクリメントするためのオフセット値を与え、前記配列の末尾に、前記挿入すべき項目値を配置し、前記添え字に、添え字変換配列中の添え字の範囲にしたがったオフセット値が与えられ、当該オフセット値が与えられた添え字により配列中の項目値が特定されることを特徴とする表形式データの挿入方法により達成される。

【0011】本発明によれば、添え字変換配列により、添え字に所定のオフセットを与えて、配列の項目値を特定している。したがって、挿入の場合に、新たに挿入された位置に対応する添え字が、添え字変換配列により、配列の末尾以降の所定の位置に配置された新たな項目値を特定することができる。したがって、実際に配列中に実際にデータを挿入し、他のデータを移動する必要がなく、論理上、配列に項目値を挿入することが可能となる。また、これにより、後述するような並列処理やトランザクション処理を適切になすことが可能となる。

【0012】また、本発明の目的は、項目とこれに含まれる項目値とを含むレコードの配列として表わされる表形式データにおいて、任意の位置の項目値を削除する表形式データの削除方法であって、レコード番号を添え字として受け入れ、受け入れられた添え字の範囲に対応するオフセット値を与えるように構成された添え字変換配列を生成し、削除すべき項目値の位置を示す削除位置を特定し、前記添え字変換配列において、削除位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を下方にシフトするとともに、受け入れられた添え字をインクリメントするためのオフセット値を与え、前記添え字に、添え字変換配列中の添え字の範囲にしたがったオフセット値が与えられ、当該オフセット値が与えられた添え字により配列中の項目値が特定されることを特徴とする表形式データの削除方法により達成される。本発明においても、添え字変換配列により、削除する位置に対応する項目値を、添え字がポイントしないように構成される。データを実際に削除し、かつ、データを移動することなく、論理上、データ削除の処理が実現される。また、表形式データの削除方法は、上記更新方法および削除方法を、順次、或いは、その逆順で実行すればよい。

【0013】また、本発明の別の実施態様においては、項目とこれに含まれる項目値とを含むレコードの配列として表わされる表形式データの項目値の値を変換する表形式データの値変換方法は、項目値の範囲に対応するオフセット値を与えるように構成された値変換配列を生成し、レコード番号を添え字として受け入れ、前記配列中の、当該添え字に対応する項目値に、前記項目値の範囲に対応したオフセット値を与えるように構成されている。また、本発明の目的は、項目とこれに含まれる項目値とを含むレコードの配列として表わされる表形式データにおいて、任意のレコード番号に対応する項目値を特定するように構成された表形式データのデータ構造であって、当該表形式データを、特定の項目に属する項目値に対応した項目値番号の順に当該項目値が格納されている第1の実体配列を含む値リストと、一意的なレコード番号の順に、当該項目値番号を指示するためのポインタ値が格納された第2の実体配列を含むポインタ配列とからなる一以上の情報ブロックに分割するように構成し、前記ポインタ配列に、入力されたレコード番号を添え字として、当該添え字の範囲に対応する第1のオフセットを与えるように構成された第1の添え字変換配列を形成するとともに、当該添え字変換配列を経てオフセットが与えられた添え字により特定される、第2の実体配列中のポインタ値を値として、当該値の範囲に対応する第2のオフセットを与えるように構成された値変換配列を形成し、前記値リストに、入力された、ポインタ配列の出力を添え字として、当該添え字の範囲に対応する第3のオフセットを与えるように構成された第2の添え字変換

配列を形成し、当該第2の添え字変換配列を経てオフセットが与えられたポインタ配列の出力により、第1の実体配列中の項目値が特定されるように構成されたことを特徴とする表形式データのデータ構造により達成される。

【0014】本発明によれば、たとえば、あるプロセスでは、添え字変換配列や値変換配列を経たデータストリームを形成し、他のプロセスでは、実体配列のみを利用するデータストリームを形成することができる。このように、必要に応じて、上記変換配列の通過や不通過を設定することにより、プロセスの並列処理やトランザクション処理を適切に実現することが可能となる。たとえば、前記添え字変換配列の各々は、所定の範囲にそれぞれ含まれる添え字の最小値を示す開始位置からなる開始位置配列、および／または、当該所定の範囲に含まれる添え字の最大値を示す終了位置からなる終了位置配列と、対応するオフセット値からなるオフセット配列とから構成されるのが好ましい。また、前記値変換配列は、所定の範囲にそれぞれ含まれる値の最小値を示す開始位置からなる開始位置配列、および／または、当該所定の範囲に含まれる値の最大値を示す終了位置からなる終了位置配列と、対応するオフセット値からなるオフセット配列とから構成されるのが好ましい。

【0015】さらに、本発明の目的は、上に定義したデータ構造にて表わされる表形式データの任意の位置に、項目値を挿入する表形式データの挿入方法であって、

(1) 前記値リストに関して、挿入される項目値の位置を示す挿入位置を特定し、前記第2の添え字変換配列において、前記挿入位置に関して、対応する添え字を確定するとともに、前記第1の実体配列の末尾以降の所定の位置を特定するための第3のオフセット値を与え、前記第2の添え字変換配列において、前記挿入位置に対応する添え字の値より大きなものに関して、対応する添え字の範囲を、当該範囲を確定する値が大きくなるように上方にシフトするとともに、受け入れられた添え字をデクリメントするための第3のオフセット値を与え、前記第1の実体配列の末尾以降の所定の位置に、前記挿入すべき項目値を配置し、(2) 前記ポインタ配列に関して、挿入されるレコード番号に対応するポインタ値の位置を示す挿入位置を特定し、前記第1の添え字変換配列において、前記挿入位置に関して、対応する添え字を確定するとともに、前記第1の実体配列の末尾以降の所定の位置を特定するための第1のオフセット値を与え、前記第1の添え字変換配列において、前記挿入位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を、当該範囲を確定する値が大きくなるように上方にシフトするとともに、受け入れられた添え字をデクリメントするための第1のオフセット値を与え、前記第2の実体配列の末尾以降の所定の位置に、現存するポインタ値より大きな新たなポインタ

値を配置し、かつ、(3)前記ポインタ配列に関して、前記値変換配列に、前記値リストにおける挿入位置に対応するポインタ値以上を有するものに対して、これをインクリメントする第2のオフセット値を与え、前記値変換配列に、前記新たなポインタ値が、前記挿入位置に対応する位置を特定するような第2のオフセット値を与えることを特徴とする表形式データのデータ挿入方法により達成される。

【0016】上記挿入方法の好ましい実施態様においては、 m 個のレコード番号と、 n 個の項目値とを備え、前記値リスト中の項目値の挿入位置が、 i ($0 \leq i \leq n-1$) であり、第1の実体配列の末尾 n に項目値が配置され、かつ、レコード番号の挿入位置が j ($0 \leq j \leq m-1$) であり、第2の実体配列の末尾 m にポインタ値が配置される場合に、(1)前記値リストの第2の添え字変換配列において、値の範囲が $(i-1)$ 以下の場合に、第3のオフセット値として0を与え、値が i である場合に、第3のオフセット値として $(n-i)$ を与え、かつ、値の範囲が $(i+1)$ 以上 n 以下である場合に、第3のオフセット値として (-1) を与え、(2)前記ポインタ配列の第1の添え字変換配列において、添え字の範囲が $(j-1)$ 以下の場合に、第1のオフセット値として0を与え、添え字が j である場合に、第1のオフセット値として $(m-j)$ を与え、かつ、添え字が $(j+1)$ 以上 m 以下の場合に、第1のオフセット値として (-1) を与え、(3)前記ポインタ配列の値変換配列において、値の範囲が $(i-1)$ の場合に、第2のオフセット値として0を与え、値の範囲が i 以上 $(n-1)$ 以下である場合に、第2のオフセット値として1を与え、かつ、値が n である場合に、第2のオフセット値として $(i-n)$ を与えるように構成されている。

【0017】或いは、 m 個のレコード番号と、 n 個の項目値とを備え、前記値リスト中の項目値の挿入位置が、 i ($0 \leq i \leq n-1$) であり、第1の実体配列の末尾以降の所定の位置 z ($z \geq n$)に項目値が配置され、かつ、レコード番号の挿入位置が j ($0 \leq j \leq m-1$) であり、第2の実体配列の末尾以降の所定の位置 x ($x \geq m$)にポインタ値が配置される場合に、(1)前記値リストの第2の添え字変換配列において、値の範囲が $(i-1)$ 以下の場合に、第3のオフセット値として0を与え、値が i である場合に、第3のオフセット値として $(z-i)$ を与え、かつ、値の範囲が $(i+1)$ 以上 n 以下である場合に、第3のオフセット値として (-1) を与え、(2)前記ポインタ配列の第1の添え字変換配列において、添え字の範囲が $(j-1)$ 以下の場合に、第1のオフセット値として0を与え、添え字が j である場合に、第1のオフセット値として $(x-j)$ を与え、かつ、添え字が $(j+1)$ 以上 m 以下の場合に、第1のオフセット値として (-1) を与えるとともに、(3)前記ポインタ配列の値変換配列において、値の範囲が

$(i-1)$ の場合に、第2のオフセット値として0を与え、値の範囲が i 以上 $(n-1)$ 以下である場合に、第2のオフセット値として1を与え、かつ、値が y (ただし、 y は、第2の実体配列の位置 x に格納されたポインタ値)である場合に、第2のオフセット値として $(i-y)$ を与えるように構成されてもよい。

【0018】また、本発明の目的は、上記定義に基づくデータ構造にて表わされる表形式データの任意の位置の項目値を削除する表形式データの削除方法であって、前記ポインタ配列に関して、削除されるポインタ値の位置を示す削除位置を特定し、前記第1の添え字変換配列において、前記削除に関して、対応する添え字を確定し、前記第1の添え字変換配列において、前記削除位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を、当該範囲を確定する値が小さくなるように下方にシフトするとともに、受け入れられた添え字をインクリメントするための第1のオフセット値を与えることを特徴とする表形式データの削除方法により達成される。上記削除方法の好ましい実施態様においては、 m 個のレコード番号と、 n 個の項目値とを備え、レコード番号の削除位置が j ($0 \leq j \leq m-1$) である場合に、前記ポインタ配列の第1の添え字変換配列において、添え字の範囲が $(j-1)$ 以下の場合に、第1のオフセット値として0を与え、かつ、添え字の範囲が j 以上 $(m-2)$ 以下である場合に、オフセット値として1を与えるように構成されている。

【0019】さらに、本発明の目的は、上記定義によるデータ構造にて表わされる表形式データの任意の位置の項目値を更新する表形式データの更新方法であって、

(A) (1)前記値リストに関して、更新される項目値を挿入される項目値と仮定し、当該項目値の挿入すべき位置を示す挿入位置を特定し、前記第2の添え字変換配列において、前記挿入位置に関して、対応する添え字を確定するとともに、前記第1の実体配列の末尾以降の所定の位置を特定するためのオフセット値を与え、前記第2の添え字変換配列において、前記挿入位置に対応する添え字の値より大きなものに関して、対応する添え字の範囲を、当該範囲を確定する値が大きくなるように上方にシフトするとともに、受け入れられた添え字をデクリメントするための第3のオフセット値を与え、前記第1の実体配列の末尾以降の所定の位置に、前記挿入すべき項目値を配置し、(2)前記ポインタ配列に関して、更新されるポインタ値を挿入されるポインタ値と仮定し、当該ポインタ値の挿入すべき位置を示す挿入位置を特定し、前記第1の添え字変換配列において、前記挿入位置に関して、対応する添え字を確定するとともに、前記第1の実体配列の末尾以降の所定の位置を特定するための第1のオフセット値を与え、前記第1の添え字変換配列において、前記挿入位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え

字の範囲を、当該範囲を確定する値が大きくなるように上方にシフトするとともに、受け入れられた添え字をデクリメントするための第1のオフセット値を与え、前記第2の実体配列の末尾以降の所定の位置に、現存するポインタ値より大きな新たなポインタ値を配置し、かつ、
 (3) 前記ポインタ配列に関して、前記値変換配列に、前記値リストにおける挿入位置に対応するポインタ値以上を有するものに対して、これをインクリメントする第2のオフセット値を与え、前記値変換配列に、前記新たなポインタ値が、前記挿入位置に対応する位置を特定する
 10 ような第2のオフセット値を与え、(B)前記ポインタ配列に関して、更新すべきポインタ値の位置を、削除位置として、前記挿入位置を考慮して特定し、前記第1の添え字変換配列において、前記削除位置に関して、対応する添え字を確定し、前記第1の添え字変換配列において、前記削除位置に対応するレコード番号より大きなレコード番号を有するものに関して、対応する添え字の範囲を、当該範囲を確定する値が小さくなるように下方にシフトするとともに、受け入れられた添え字をインクリメントするための第1のオフセット値を与え、前記(A)
 20 および(B)、或いは、前記(B)および(A)の何れかを順次実行するように構成されたことを特徴とする表形式データの更新方法により達成される。

【0020】上記更新方法の好ましい実施態様においては、 m 個のレコード番号と、 n 個の項目値とを備え、前記値リスト中の項目値の更新位置が、 i ($0 \leq i \leq n-1$) であり、第1の実体配列の末尾 n に項目値が配置され、かつ、レコード番号の更新位置が j ($0 \leq j \leq m-1$) であり、第2の実体配列の末尾 m にポインタ値が配置される場合に、(1) 前記値リストの第2の添え字変換配列において、値の範囲が $(i-1)$ 以下の場合に、第3のオフセット値として0を与え、値が i である場合に、第3のオフセット値として $(n-i)$ を与え、かつ、値の範囲が $(i+1)$ 以上 n 以下である場合に、第3のオフセット値として (-1) を与え、(2) 前記ポインタ配列の第1の添え字変換配列において、添え字の範囲が $(j-1)$ 以下の場合に、第1のオフセット値として0を与え、添え字が j である場合に、第1のオフセット値として $(m-j)$ を与え、かつ、添え字が $(j+1)$ 以上 $(m+1)$ 以下の場合に、第1のオフセット値として0を与え、(3) 前記ポインタ配列の値変換配列において、値の範囲が $(i-1)$ の場合に、第2のオフセット値として0を与え、値の範囲が i 以上 $(n-1)$ 以下である場合に、第2のオフセット値として1を与え、かつ、値が n である場合に、第2のオフセット値として $(i-n)$ を与えるように構成されている。

【0021】或いは、 m 個のレコード番号と、 n 個の項目値とを備え、前記値リスト中の項目値の更新位置が、 i ($0 \leq i \leq n-1$) であり、第1の実体配列の末尾以降の所定の位置 z ($z \geq n$) に更新された項目値が配置

され、かつ、レコード番号の更新位置が j ($0 \leq j \leq m-1$) であり、第2の実体配列の末尾以降の所定の位置 x ($x \geq m$) に更新されたポインタ値が配置される場合に、(1) 前記値リストの第2の添え字変換配列において、値の範囲が $(i-1)$ 以下の場合に、第3のオフセット値として0を与え、値が i である場合に、第3のオフセット値として $(z-i)$ を与え、かつ、値の範囲が $(i+1)$ 以上 n 以下である場合に、第3のオフセット値として (-1) を与え、(2) 前記ポインタ配列の第1の添え字変換配列において、添え字の範囲が $(j-1)$ 以下の場合に、第1のオフセット値として0を与え、添え字が j である場合に、第1のオフセット値として $(x-j)$ を与え、かつ、添え字が $(j+1)$ 以上 $(m+1)$ 以下の場合に、第1のオフセット値として0を与えるとともに、(3) 前記ポインタ配列の値変換配列において、値の範囲が $(i-1)$ の場合に、第2のオフセット値として0を与え、値の範囲が i 以上 $(n-1)$ 以下である場合に、第2のオフセット値として1を与え、かつ、値が y (ただし、 y は、第2の実体配列の位置 x に格納されたポインタ値) である場合に、第2のオフセット値として $(i-y)$ を与えるように構成してもよい。

【0022】或いは、上記定義にしたがった表形式データの削除方法は、前記レコード番号に関して、レコード番号自体を配置した第4の実体配列と、当該レコード番号を特定するための添え字の範囲にしたがって、所定のオフセット値を与える第3の添え字変換配列とを設け、前記削除すべきレコード番号の位置を示す削除位置を特定し、前記第3の添え字変換配列において、前記削除に関して対応する添え字を確定し、前記第3の添え字変換配列において、前記削除位置に対応する添え字より大きな値を有するものに関して、対応する添え字の範囲を、当該範囲を確定する値が小さくなるように下方にシフトするとともに、受け入れられた添え字をインクリメントするための第4のオフセット値を与えるように構成してもよい。これは、レコード番号自体を配置した第4の実体配列を含む情報ブロックを形成し、レコード番号を指定するための添え字変換配列を形成し、これにより実質的なデータの削除を実現している。

【0023】また、本発明の目的は、上記方法により表形式データにおけるデータの挿入、削除および/または更新をなし、ロールバックおよびコミットの何れかの場合には、前記添え字変換配列および値変換配列を廃棄することと特徴とするトランザクション処理方法によっても達成される。この場合に、ロールバックの場合にのみ、前記添え字変換配列および値変換配列を廃棄してもよい。また、本発明の目的は、上記定義にしたがったデータ構造を有する表形式データに関して、複数のプロセスにより同時に処理が実行されるプロセスの並列処理方法であって、前記プロセスのうち、データの挿入、削除

および/または更新を伴うものは、前記ポインタ配列において、第1の添え字変換配列、第2の実体配列および値変換配列を経るよう構成されるとともに、前記値リストにおいて、第2の添え字変換配列および第1の実体配列を経るよう構成され、その一方、前記プロセスのうち、データの挿入、削除および/または更新を伴わないものは、前記ポインタ配列において、第2の実体配列を経るとともに、前記値リストにおいて、前記値リストにおいて、第1の実体配列を経るよう構成されたことを特徴とするプロセスの並列処理方法によっても達成される。データの挿入、削除および/または更新を伴うプロセスは、上述したような挿入方法、削除方法および更新方法の何れかを利用しているのが好ましい。

【0024】或いは、本発明のさらに別の実施態様においては、上記データ構造を有する表形式データに関して、レコードのロックをなす方法は、ロックを管理する配列を含む情報ブロックを設け、前記情報ブロックの配列中に、前記レコード番号の各々に対応して、ロックの種別を示す項目値を配置するよう構成されている。また、本発明の目的は、上記種々の方法の少なくとも何れかを実現するプログラムを記憶した、コンピュータにより読み出し可能な記憶媒体によっても達成される。

【0025】

【発明の実施の形態】以下、添付図面を参照して、本発明の実施の形態につき説明を加える。図1は、本発明の実施の形態にかかる表形式データにかかる、ある項目のデータを挿入、削除および更新する方法を実現できるコンピュータシステムのハードウェア構成を示すブロックダイアグラムである。図1に示すように、このコンピュータシステム10は、通常のものと同様の構成であり、プログラムを実行することにより、システム全体および個々の構成部分を制御するCPU12、ワークデータなどを記憶するRAM(Random Access Memory)14、プログラム等を記憶するROM(Read Only Memory)16、ハードディスク等の固定記憶媒体18、CD-ROM19をアクセスするためのCD-ROMドライブ20、CD-ROMドライブ20と外部ネットワーク(図示せず)と接続された外部端子との間に設けられたインタフェース(I/F)22、キーボードやマウスからなる入力装置24、CRT表示装置26を備えている。CPU12、RAM14、ROM16、外部記憶媒体18、I/F22、入力装置24および表示装置26は、バス28を介して相互に接続されている。

【0026】本実施の形態にかかる表形式データにかかるある項目のデータを挿入するプログラム、ある項目のデータを削除するプログラム、ある項目のデータを更新するプログラム、トランザクション処理を実現するプログラム、並列処理を司るプログラム、および、排他処理を実現するプログラムは、CD-ROM19に収容され、CD-ROMドライブ20に読取られても良いし、

ROM16に予め記憶されていても良い。また、いったんCD-ROM19から読み出したものを、外部記憶媒体18の所定の領域に記憶しておいても良い。或いは、上記プログラムは、ネットワーク(図示せず)、外部端子およびI/F22を経て外部から供給されるものであっても良い。なお、本明細書において、データの挿入とは、隣接するレコードの間にデータを入れることのみならず、レコードの末尾に新規のレコードを追加することを含む。

【0027】また、本実施の形態においては、データの挿入、削除および更新処理、並びに、トランザクション処理、並列処理、排他処理を適切かつ高速に実現するために、後述するように、表形式データに基づき、所定のデータ形式の情報ブロックを生成する必要がある。この情報ブロック生成プログラムも同様に、CD-ROM19に収容され、ROM16に記憶され、或いは、外部記憶媒体18に記憶されても良い。或いは、これらプログラムは、ネットワーク(図示せず)を介して、外部から供給されても良いことはいうまでもない。また、本実施の形態において、情報ブロック生成プログラムにて生成されたデータ(情報ブロック)は、RAM14に記憶され、或いは、外部記憶媒体18の所定の領域に記憶される。

【0028】まず、本発明の前提となる表形式データの取扱い、すなわち、表形式データを管理する形態につき簡単に説明を加える。本発明者は、表形式データを高速に検索、集計およびソートを高速に実現するため、また、複数の表形式データを所望のようにジョインできるようにするため、特定のデータ形式を有する表形式データの構築(特願平10-227278号)と、当該特定のデータ形式を有する表形式データのジョイン(特願平11-151156号)とを考案している。本発明においても、基本的には、これら出願に開示された技術に基づき、表形式データを所定の情報ブロックの集合体として構築し、これを用いて検索、集計およびソートを実現している。図2は、本実施の形態にて用いる情報ブロックを示す図である。図2に示すように、情報ブロック100は、値リスト110と値リストへのポインタ配列120とを含んでいる。値リスト110は、表形式データの各項目に対して、その項目に属する項目値が順序付け(整数化)された項目値番号の順番に、上記項目値番号に対応した項目値111が格納されたテーブルである。値リストへのポインタ配列120は、表形式データの列(すなわち項目)の項目値番号、つまり値リスト110へのポインタが表形式データのレコード番号順に格納された配列である。

【0029】上記値リストへのポインタ配列120と値リスト110とを組み合わせることにより、あるレコード番号が与えられたときに、所定の項目に関する値リストへのポインタ配列120からそのレコード番号に対応

して格納された項目値番号を取り出し、次いで、値リスト110内でその項目値番号に対応して格納された項目値を取り出すことにより、レコード番号から項目値を得ることができる。したがって、従来のデータ表と同様に、レコード番号(行)と項目(列)という座標を用いてすべてのデータ(項目値)を参照することができる。たとえば、図3(a)に示す表形式データを考える。この例では、顧客ID、顧客名、電話番号という項目に種々の項目値が与えられている。本実施の形態においては、このような表形式データを、図3(b)ないし

(d)に示す形式の情報ブロックとして保持している。たとえば、図3(b)において、ポインタ配列120-1は、顧客IDを示す項目値を格納した値リスト110-1に関連付けられている。すなわち、先頭レコード(レコード番号「0」)のポインタ配列のポインタ値は0であり、これに対応して、顧客IDを示す項目値「1」が得られる。図3(c)において、ポインタ配列120-2は、顧客名を示す項目値を格納した値リスト110-2に関連付けられている。たとえば先頭レコード(レコード番号「0」)のポインタ配列におけるポ

インタ値は「5」であり、これに対応して、顧客名を示す項目値「山田 〇男」が得られる。図3(d)においても、同様に、ポインタ配列120-3が、電話番号を示す項目値を格納した値リスト110-3に関連付けられていることが理解できよう。また、各値リストにおいては、項目値が順序付けられて(この例では昇順)いることが理解できよう。

【0030】さらに、本実施の形態においては、情報ブロック100の値管理テーブルは、値リスト110のほか、検索や集計のために用いる分類番号フラグ配列、項目値に対応するポインタを格納すべきメモリ空間の先頭アドレスを示す開始位置配列、および、存在数配列が含まれている。分類番号フラグ配列の各フラグ、および、存在数配列の各存在数は、項目値の各々に対応付けられている。分類番号フラグのフラグ値は、通常「0」であり、検索や集計の際に見出すべき項目値に対応して「1」にセットされる。また、存在数は、その項目値を有するレコードの個数に対応する。なお、開始位置は、対応するポインタ値よりも小さなポインタ値に対応する存在数を加算したのに対応するため、必ずしも設ける必要はない。図4(a)は、表形式データの他の例を示す図、図4(b)および(c)は、それぞれ、「性別」および「年令」に関する情報ブロックを示す図である。図4(b)に示すように、性別に関する情報ブロック200-1の値管理テーブル210-1には、ポインタ配列220の各ポインタ値に対応する項目値(「男性」および「女性」)と、各項目値に対応する分類番号、開始位置および存在数が示されている。たとえば、ポインタ値が「0」(つまり、値リストの項目値が「男性」)であるようなレコードの数は632564個であり、その

一方、ポインタ値が「1」(つまり、値リストの項目値が「女性」)であるようなレコードの数は367436個となっている。また、各項目値に対応する開始位置は、後述するレコードへのポインタ配列230-1の先頭アドレスを示している。図4(c)においても、同様のことが理解できよう。

【0031】このようなデータ構造を有する情報ブロックを用いた検索の一例および情報ブロックの生成処理につき、以下に説明する。図5は、単一項目に関する検索手法を示すフローチャートである。この処理は、CPU12(図1参照)が所定の検索プログラムを実行することにより実現される。この例では、「年令」の項目値が16歳または19歳であるレコードが検索される。まず、表形式データに関する情報ブロックのうち、図4(c)に示す「年令」に関する情報ブロック200-2が特定される(ステップ501)。

【0032】次いで、特定された情報ブロック(以下、「特定情報ブロック」と称する。)の値リスト210-2において、項目値が上記検索条件に合致するもの(16歳または19歳)に対応する行の分類番号が「1」にセットされる(ステップ502)。本例の場合には、項目値番号「0」および項目値番号「3」に対応する行の分類番号が1にセットされる。次いで、分類番号が「1」にセットされている行に対応した開始位置および存在数が取得される(ステップ503)。これら情報をポインタ取り出し情報と称する。レコードへのポインタ配列において、ステップ503にて取得されたポインタ取り出し情報に基づき、検索条件に合致したレコードへのポインタを示すレコード番号が取り出される(ステップ504)。本例においては、項目値番号「0」に対応したレコードのポインタは、レコードへのポインタ配列の開始位置「0」すなわち先頭から、45898個目までの領域に格納され、その一方、項目値番号「3」に対応したレコードのポインタは、レコードへのポインタ配列の2383137番目から189653個分の領域に格納されていることがわかる。最後に、後の処理にて利用できるようにするために、取り出されたレコード番号の配列が、結果集合として作成され、これが保持される(ステップ505)。また、集計およびソートも、分類番号、開始位置および存在数を利用することにより実現することができる。

【0033】次に、上述したような検索処理等に利用するための情報ブロックの生成処理につき説明を加える。図6は、表形式データに基づき情報ブロックを作成するための処理を説明するフローチャートである。まず、システム10は、表形式の原データを取得し、これを項目別のものに分解する(ステップ601)。この原データは、たとえば、図7(a)に示すものでも良いし、或いは、図7(b)に示すものでも良い。これら原データは、外部から供給されるものであっても良いし、或い

は、固定記憶媒体18に記憶されたものであっても良い。以下に述べるステップ602ないしステップ604からなる処理ブロック610は、ある一つの項目に関する情報ブロックの生成を示す。したがって、複数の項目に関する情報ブロックを生成する場合には、項目の数だけ処理ブロック610に対応する処理が実行される。以下、「性別」に関する項目の情報ブロックを例にとって説明を加える。

【0034】まず、「性別」に関する項目の情報ブロック用の領域が、たとえば、RAM14中に確保される（ステップ602）。次いで、この確保された領域中に、値管理テーブルが生成される。より詳細には、まず、値管理テーブルが初期化される。次いで、原データのうち、「性別」に関するデータを先頭から末尾まで走査することにより、どのような項目名が、それぞれいくつ存在するかが見出される。本例では、「女性」および「男性」という項目名が、それぞれ、367436個および632564個だけあることが見出される。これにより、値リストに、「女性」および「男性」という項目値がセットされ、また、存在数配列にも所定の数がセットされる。その後、項目値が所定の基準にしたがってソートされる。ソートの際には、項目数の並び替えにしたがって存在数も並びかえられる。次いで、開始位置配列の値が決定される。これは、ソートにより自己より上位に位置する存在数を累算することにより求められる。また、開始位置配列の値を、対応する分類番号配列の値に割り当てる。この値は次のステップにて用いられる。このようにして値管理テーブルが生成された後に、レコードへのポインタ配列が生成される。このポインタ配列の領域の大きさは、存在数の総和に対応する。このようにして、所定の項目に関する情報ブロックを作り出すことが可能となる。この情報ブロックの生成を予め行っておき、生成された情報ブロックを用いて検索、集計およびソートの処理が実行される。

【0035】次に、上記形態の情報ブロックにおいて、要素を挿入、削除あるいは更新する処理につき、以下に説明を加える。たとえば、100万レコード（レコード番号0～999,999）のサイズを持つ配列の先頭から2番目に、1つの要素を挿入することを考える。たとえば、レコード番号0～999,999に対応する配列のうち、レコード番号「0」と「1」との間に、新たに一つの要素を挿入する場合である（図8（a）参照）。この場合に、新たなレコード番号「1」に新たな値（たとえば「10」が挿入され）、これ以降の値（すなわち999,999個の値）を一つずつ移動しなければならない。また、特定の要素を削除する場合にも、莫大な数の要素を移動する必要がある。或いは、上記配列において、特定の値を持つ要素を別の値に書きかえる場合、たとえば、1以上の値を有するものをインクリメントする場合（図8（b）参照）にも、配列全体を走査する必要

が有り、処理に莫大な時間を要する。ところで、上記配列は、添え字が配列中の位置を示すことにより、対応する位置の格納値を取り出す（すなわち添え字から格納値を引用する）ことができる点で有利である。つまり、各要素（格納値）がメモリのアドレス順に格納されていれば、添え字から直ちにアドレスを算出し、対応するアドレスの拡張値を得ることができる。そこで、この利点を生かしつつ、要素（格納値）の挿入および削除に要する負荷を最小にする手法が求められる。

10 【0036】本発明においては、図9（a）に示すような、添え字を配列に与え、対応する値（格納値）を取り出すような系において、図9（b）ないし（d）に示すように、配列に与える「添え字」を変換する添え字変換部、および／または、配列から取り出された値を変換する値変換部を設けるとともに、配列自体を、実体配列と差分配列とに分割し、配列に含まれる要素（格納値）の位置を変更することなく、かつ、値自体を変更することなく、要素の挿入、削除および更新を実現可能にしている。

20 【0037】[添え字変換配列による要素の挿入] まず、本発明の第1の実施の形態にて利用される、要素の挿入等のための添え字変換配列につき、説明を加える。図2ないし図4に示す情報ブロックに関して考えると、添え字には、値リストへのポインタ配列の要素（ポインタ値）を指定するためのレコード番号や、値リストの項目値を指定するためのポインタ配列中のポインタ値等が含まれる。また、配列には、値リストへのポインタ配列、値リスト、レコード番号へのポインタ配列等が含まれる。たとえば、図10（a）に示すように、配列の第30 0行（添え字「0」）と第1行（添え字「1」）との間に、要素「Y0」を挿入する場合を考える。この場合には、要素「Y0」の挿入後には、添え字と配列中の要素（格納値）とは、図10（b）に示す論理的関係となる。

【0038】図10（b）に示す論理的関係を保持しつつ、挿入時に生じる負荷を最小限にするために、図10（c）に示す添え字変換配列を考える。図10（c）において、添え字変換配列は、開始位置配列と、終了位置配列と、オフセット配列とからなる。これらの配列に格納された要素（値）のうち、同じ位置（たとえば第0 40 行、第1行等）に配置されたものが相互に関連している。開始位置配列中の要素と、終了位置配列中の対応する要素により、関連する添え字の範囲が特定される。すなわち、開始位置配列中の要素は、関連する添え字の最小値を示し、終了位置配列中の要素は、関連する添え字の最大値を示す。その一方、オフセット配列中の対応する要素は、添え字に加減算して、実際の実体配列を指定する変換済の添え字を得るためのオフセット値を示す。なお、本明細書において、開始位置配列、終了位置配列およびオフセット配列の対応する要素の組（すなわち同

じ行に位置する要素の組)を、場合によって「構造体配列」と称し、各配列中の要素を「構造体のメンバー」と称する。この添え字変換配列の意味するところにつき以下に説明する。上述したように、ある構造体配列中の構造体のメンバーにより、添え字の範囲および当該範囲に含まれる添え字のオフセット値が決定される。このため、ある添え字に対して、その添え字に関連する構造体配列を特定し、オフセット値を加えることにより、変換済の添え字を得ることができ、これを使用して配列(実体配列)の位置を特定して、配列中の要素(格納値)を参照することが可能となる。

【0039】たとえば、図10(c)において、添え字「0」は、第0行(1行目)の構造体配列と関連することが理解できる。この構造体配列において、オフセット配列中のメンバーの値(オフセット値)は「0」であるため、変換済の添え字は、「0+0」=0となる。したがって、もともとの添え字「0」に対応する実体配列中の要素は、「0」の位置に格納されていることが分かる。また、添え字「1」は、第1行(2行目)の構造体配列と関連し、当該構造体配列に関するオフセット配列中のメンバーの値(オフセット値)は、「n-1」であるため、変換後の添え字は、「1+(n-1)=n」となる。したがって、対応する要素は、実体配列の「n」の位置に格納されていることが理解できる。添え字「2」ないし「n」に関しても同様に、対応する要素の格納されている位置が特定できることは明らかである。

【0040】より詳細に、上記手順につき、図11のフローチャートを参照して説明を加える。まず、添え字の位置を示す番号(格納位置番号)が初期化され(ステップ1101)、次いで、開始配列および終了配列の要素を参照して、添え字が、開始位置配列および終了位置配列の値により画定される範囲のうち、何れに含まれるかが特定される(ステップ1102)。すなわち、このステップにより、添え字がどの構造体配列に関連するかが特定される。たとえば、図10(c)の例において、添え字「0」は、開始配列中の要素「0」および終了配列の要素「0」にて特定される範囲に含まれることが分かる。或いは、添え字「1」は、開始配列中の要素「1」および終了配列中の要素「1」にて特定される範囲に含まれることがわかる。

【0041】次いで、その範囲に基づき、対応するオフセット値が取り出され(ステップ1103)、添え字にオフセット値を加えた値が得られる(ステップ1104)。このオフセット値が加えられた値が、変換済の添え字となるため、この値を利用して、実体配列中の要素が特定される(ステップ1105)。たとえば、添え字「1」に関して、この添え字は第1行(2行目)に対応する範囲に含まれるため、第1行のオフセット値である「n-1」が添え字「1」に加えられる。したがって、

変換済の添え字は「n」となり、実体配列中の対応する要素(格納値)「Y0」を得ることができる。或いは、添え字「2」に関して、この添え字は、第2行(3行目)に対応する範囲に含まれるため、第2行のオフセット値である「-1」が添え字「2」に加えられる。したがって、変換済の添え字は「1」となり、実体配列中の対応する要素(格納値)「X1」を得ることができる。必要な添え字につきステップ1102ないし1105の処理を繰り返すことにより(ステップ1106および1107参照)、図10(b)に示す論理的関係を維持しつつ、実体配列中の要素(格納値)を得ることができる。したがって、本発明においては、配列中の要素(格納値)の挿入時には、上述したような変換配列を生成するとともに、配列中の要素自体を移動させることなく、実体配列の所定の位置(たとえば、アドレスの最大値に対応する位置)に、挿入すべき要素を配置することにより、負荷が生じることなく、要素の挿入を実現できるようにしている。

【0042】[要素挿入のための添え字変換配列の生成]次に、第1の実施の形態にかかる上記添え字変換配列の作成手順につき説明を加える。たとえば、図12に示すように、要素数が5個の配列(符号1201)を考え、これのうち、第0行(1行目)と第1行(2行目)との間に、新たな要素(格納値)「Y0」を挿入する場合を考える(符号1202参照)。図13は、配列に新たな要素(格納値)を挿入するための変換配列の生成処理を説明するフローチャートである。この処理は、図1におけるCPU12により実行される。まず、初期状態の添え字変換配列が生成される(ステップ1300)。より詳細には、配列の要素数が調べられ、開始位置および終了位置が画定される。たとえば、図12に示す例では、要素数が「5」であるため、初期的な添え字変換配列において、開始位置が「0」であり、終了位置が「4」となる。また、この例では、添え字から直接、配列中の要素が参照されるようになっているため、初期的なオフセット値が「0」となる(図14の符号1401参照)。なお、添え字変換配列が予め生成されている場合、すなわち、既に一度以上の挿入がなされている場合や、情報ブロック生成時に初期的な添え字変換配列が生成されている場合には、ステップ1300は省略される。

【0043】次いで、挿入すべき要素の位置が特定され(ステップ1301)、挿入位置、挿入位置の前および挿入位置の後に係る添え字変換配列のための領域が確保される。図14の符号1401の例では、初期的な添え字変換配列に含まれる各配列が、それぞれ1つの要素のみを含んでいたが、ステップ1301の処理により、3つの要素(挿入位置の前に関する要素、挿入位置に関する要素、および、挿入位置の後に係る要素)を有する配列のための領域が確保される(ステップ1302)

2)。なお、既に各々が複数の要素を有する配列が設けられていた場合には、要素数が2つ増えるような配列のための領域が確保される。

【0044】その後、挿入位置より前に関して、各配列の要素の値が決定される(ステップ1303)。より詳細には、開始位置配列の対応する要素値およびオフセット配列の対応する要素値は変更されず、終了位置配列の要素値として、「挿入位置に対応する値-1」が与えられる。また、挿入位置に関しても、各配列の対応する要素の値が決定される(ステップ1304、1305)。この場合には、開始位置配列および終了位置配列の要素値として、「挿入位置に対応する値」が与えられ、かつ、オフセット配列の要素値として、「実体配列の末尾の位置に対応する値」が与えられる。なお、後述するように「差分配列」として、挿入すべき要素を別個の位置に配置するのであれば、当該差分配列中の、挿入すべき要素の配置される位置に対応する値をオフセット値として与えれば良い。また、挿入位置より後に関しても、以下の手法により各配列の要素値が決定される。すなわち、開始位置配列の対応する要素値として、「挿入位置に対応する値+1」が与えられ(ステップ1306)、終了位置配列の対応する値として、初期的な「終了位置に対応する値+1」が与えられる(ステップ1307)。次いで、オフセット配列の対応する要素値が、「その初期値(元の値)-1」に設定される(ステップ1308)。

【0045】たとえば、図12の符号1201に示すような添え字および配列があり、ステップ1300により、図14の符号1401に示す初期的な添え字変換配列が生成された場合には、上記ステップ1301ないしステップ1307の処理により、符号1411に示す添え字変換配列が生成されることが理解できよう。このような添え字変換配列を生成した後、CPU12は、ステップ1304にて設定された実体配列の末尾以降の所定の位置(上記例では末尾)に、挿入すべき要素を格納する。これにより、実体配列の要素(格納値)を移動することなく、実質的に(論理上)実体配列の所望の位置に要素を挿入することが可能となる。

【0046】[添え字変換配列による要素の削除]次に、上述した添え字変換配列を用いて、要素を削除する手法につき説明を加える。たとえば、図15(a)に示すように、配列の第1行(2行目)の要素を削除する場合を考える。この場合には、削除の後に、添え字と配列中の要素(格納値)とは、図15(b)に示す論理的関係となる。挿入の場合と同様に、図15(b)に示す論理的関係を保持しつつ、削除時に生じる負荷を最小限にするために、図15(c)に示す添え字変換配列を考える。図15(c)においても、図10(c)と同様に、構造体配列により添え字の範囲およびオフセット値が規定され、変換済の添え字が求められるようになってい

る。より詳細には、添え字の各々に関して、図11の処理を施すことにより、実体配列中の要素(格納値)が特定されることが理解できる。たとえば、添え字「1」は、第1行(2行目)の構造体配列と関連し、当該構造体配列に関するオフセット配列中のメンバーの値(オフセット値)は「1」であるため、変換後の添え字は、「1+1=2」となる。したがって、対応する要素は、実体配列の「2」に対応する位置に格納されていることが理解できる。

【0047】[要素削除のための添え字変換配列の生成]たとえば、図16(a)に示すように、5個の要素からなる配列(符号1601)を考え、これのうち、第1行(2行目)の要素(格納値)X0を削除する場合を考える(符号1602参照)。なお、この例では、実体配列は、配列1601と同じであると仮定する。図17は、配列から要素(格納値)を削除するための変換配列の生成処理を示すフローチャート、図18(a)は、図16(a)に示す例に関して、処理に伴って変換配列中の数値が変化する状態を示す図である。図17に示す処理も、図1におけるCPU12により実行される。まず、添え字変換配列が生成されていない場合には、初期状態の添え字変換配列が生成される(ステップ1700)。たとえば、図16(a)に示す例では、要素数が「5」であるため、その開始位置を示す値が「0」、終了位置を示す値が「4」、オフセット値が「0」となるような開始位置配列、終了位置配列およびオフセット配列が生成される(図18(a)の符号1801参照)。

【0048】次いで、削除すべき要素の位置が特定される(ステップ1701)、当該削除位置に関する変換配列のための領域が特定される(ステップ1702)。図18(a)に示す例では、符号1802に示すように、変換配列中の各配列において、削除位置の前、削除位置および削除位置の後に係る要素を収容するための領域が選ばれる。新たな領域において、削除位置の前に関し、終了位置配列の要素として、「削除位置を示す値-1」が配置され(ステップ1703)、その一方、削除位置の後に係る、開始位置配列の要素として、「削除位置を示す値+1」が配置される(ステップ1704)。このようにして、削除位置を含む変換配列中の構造体配列に関して、符号1802に示すような開始位置配列、終了位置配列およびオフセット配列が作成される。なお、削除位置に関する構造体配列は、最終的には使用されないため、削除位置自体を算出することは必要であるが、図18(a)に示すように、要素を配列中に格納する必要はない。

【0049】次いで、各配列に関して、削除位置より後の要素に関して、以下に述べる演算が施される。まず、開始位置配列の要素が「もとの値-1」に設定され(ステップ1705)るとともに、終了位置配列の要素が「もとの値-1」に設定される(ステップ1706)。

また、オフセット配列の要素が「もとの値+1」と設定される(ステップ1707)。たとえば、図18(a)の例において、削除位置より後ろに位置する開始位置配列中の要素「2」がデクリメントされ、「1」となり(符号1804参照)、終了位置配列中の要素「4」がデクリメントされ「3」となり、かつ、オフセット配列中の要素「0」がインクリメントされ「1」となる。このようにして、配列中の要素を削除するための添え字変換配列を得ることが可能となる。なお、実体配列(すなわち、添え字にて特定される要素を実際に格納した配列)は、要素の削除の際にも変更されない。すなわち、添え字変換配列のみによって、見かけ(論理上)の配列中の要素の削除を実現している。

【0050】また、図16(b)に示す他の例についても、図17の処理により配列中の要素の削除が実現できる。この例では、先に説明した要素の挿入により、配列には、第3行(4行目)に要素「Y0」が挿入されており、したがって、その実体配列は、末尾に要素「Y0」が付加された形態となっている(図16の符号1610参照)。また、上記挿入の際に、図18(b)の符号1811にて示す添え字変換配列が生成されている。このような場合には、ステップ1700に示す初期変換配列の生成は省略される。図18(b)の符号1811にて示す添え字変換配列に対して、ステップ1701ないしステップ1707の処理を施すことにより、符号1813にて示す添え字変換配列が求められることが理解できよう。また、このような添え字変換配列を用いることで、実体配列(図16の符号1610)を変更することなく、添え字から変換配列を経た添え字(変換済の添え字)を用いて、図16の符号1612に示す論理関係を持たせることができることが理解されよう。

【0051】[要素の挿入および削除の結合]配列中の要素の挿入および削除は、上記図13に示す挿入処理および図17に示す削除処理を順次実行することにより実現される。上記処理の順序は逆でも容易なことは言うまでもない。たとえば、初期的に、図19に示すように、「0」ないし「4」の添え字に対して、それぞれ、「0」、「100」、「200」、「300」および「400」という要素が特定されるようになっている配列1901を考える。この配列に関して、第2行(3行目)と第3行(4行目)との間に、要素「201」を挿入するとともに、第1行(2行目)の要素を削除して、符号1902に示す配列を得ることを考える。本実施の形態においては、図20(a)に示すように、まず、初期的な変換配列を作成し(符号2001参照)、次いで、図13に示す処理によって挿入のための変換配列(図20(b)の符号2002参照)を作成するとともに、実体配列の末尾に挿入すべき要素(この場合には「201」)を配置する(符号2003参照)。次いで、図17に示す処理によって削除のための変換配列

(図20(c)の符号2004参照)を作成する。これにより、要素の挿入および削除が実現される。

【0052】なお、要素の削除、および、当該削除した要素の位置に新たな要素を挿入する処理を順次実行することにより、要素の更新が実現されることが理解できるであろう。図20(a)~(c)の右側の図表(符号2011~2013)において、格納位置は、添え字変換配列を経た添え字、格納値は、見かけ(論理的な)実体配列を示している。初期的には、添え字と格納位置とは一致している(符号2011参照)が、挿入および削除により、添え字変換配列が生成ないし更新され、見かけの実体配列において所望のように要素が挿入および削除されていることが理解できよう(符号2012、2013参照)。

【0053】[添え字変換配列の他の例]次に、第2の実施の形態にかかる添え字変換配列のサイズを削減した例につき説明を加える。前記第1の実施の形態においては、開始位置配列と終了位置配列とを用いて、添え字がどの範囲に含まれるか、或いは、削除または挿入すべき要素の位置が特定されるようになっていた。しかしながら、終了位置配列のある要素の値が「n」である場合に、次の行に位置する開始位置配列の要素の値は、「n+1」であることは明らかであり、その逆もまた明らかである。そこで、第2の実施の形態においては、添え字変換配列を、開始位置配列およびオフセット配列、或いは、終了位置配列およびオフセット配列から構成し、添え字変換配列に要するメモリサイズを削減した。たとえば、図21(a)に示す添え字変換配列2101を、図21(b)のように表わしても良い。図21(b)においては、 $0 \leq \text{添え字 } i \leq (3-1)$ であるような添え字「i」は、第0行(1行目)に対応するため、このような添え字のオフセット値は「0」となる。また、 $3 \leq \text{添え字 } i \leq (4-1)$ であるような添え字「i」(=3)は、第1行(2行目)に対応し、 $4 \leq \text{添え字 } i$ であるような添え字「i」は、第2行(3行目)に対応する。【0054】或いは、図21(c)に示すような添え字変換配列を用いても良いことは明らかである。この場合に、 $5 \geq \text{添え字 } i \geq (3+1)$ であるような添え字「i」は、第2行(3行目)に対応し、 $3 \geq \text{添え字 } i \geq (2+1)$ であるような添え字「i」は、第1行(2行目)に対応し、また、 $2 \geq \text{添え字 } i$ であるような添え字「i」は、第0行(1行目)に対応する。なお、図21(b)に示すような添え字変換配列を用いれば添え字の総数を知ることができ、これにより、配列のサイズを知ることができるため、この添え字変換配列を用いるのがより好ましい。

【0055】[実体配列と差分配列]前記第1の実施の形態においては、配列に要素を挿入する際に、配列の末尾に挿入すべき要素を配置している(たとえば、図10(c)参照)。しかしながら、オフセット配列中の対応

するオフセット値を設定することにより、所望の位置に、挿入すべき要素を配置できることは明らかである。第3の実施の形態では、この挿入される要素を、任意の編集作業領域に配置している。このような、挿入される要素が配置された編集用の作業領域を「差分配列」と称する。図22は、添え字変換配列、実体配列および差分配列の一例を示す図である。図22に示すように、この例では、添え字変換配列2201により、挿入された要素「Y0」および「Y1」が、実体配列2202とは別個の領域に確保された差分配列2203中に配置されている。これは、添え字変換配列を作成（図13参照）した後に、挿入すべき要素（格納値）を配置する際に、挿入すべき要素のオフセット値にしたがった領域に、差分配列を作成し、当該差分配列中に上記要素を配置すれば良い。

【0056】〔値変換配列〕次に、本発明の第4の実施の形態にかかる値変換配列につき説明を加える。配列に関しては、上記要素の挿入および削除のほか、配列の要素（格納値）自体を変化させる（更新する）必要が生じ得る。本実施の形態においては、配列の要素を更新するために、以下に述べる値変換配列を利用する。たとえば、図23（a）に示すように、ある配列（レコード番号リスト）のうち、1以上の値を有するものをインクリメントする必要がある場合を考える。従来の手法では、配列中の要素（格納値）が1以上であるかを判断し、1以上である場合には、要素（格納値）をインクリメントするという処理を、全ての要素に関して実行していた。これに対して、本発明の第4の実施の形態においては、配列の後段（下流側）に、値変換配列を配置し、配列から出力された値を、値変換配列を通すことにより、交換済の値を出力し、この交換済の出力において、要素（格納値）の更新が実現されているような手法を用いている。図23（a）に示すような場合には、図23（b）に示すような値変換配列が設けられる。この実施の形態では、第2の実施の形態と同様に、値変換配列は、終了位置配列とオフセット配列とから構成されている。無論、第1の実施の形態のように、値変換配列を開始位置配列、終了位置配列およびオフセット配列から構成しても良い。

【0057】これらの配列に格納された要素（値）のうち、同じ位置（たとえば第0行、第1行）に配置されたものが相互に関連している。したがって、対応する位置に配置された要素により、構造体配列を構成する。本実施の形態においては、終了位置配列の値により、実体配列中の要素（格納値）の配置されている位置が、どの構造体配列に関連しているかが特定され、それに基づき、値に対するオフセットを特定することが可能となる。上述した例（1以上の要素をインクリメントする例）に関する処理につき、図24を用いてより詳細に説明する。この処理も、先に述べた実施の形態と同様に、図1に示

すCPUにより実行される。

【0058】まず、要素の格納位置を示す番号（格納位置番号）が初期化され（ステップ2401）、当該格納位置番号により特定される配列中の要素（格納値）が、終了位置配列により画定される範囲のうち、何れに含まれるかが特定される（ステップ2402）。このステップにより、要素がどの構造体配列に関連するかが特定される。図23（b）の例において、格納位置番号が「0」（すなわち、第0行（1行目）の）要素は「1」であり、この要素は、終了位置配列中の第1行（2行目）の値にて確定される範囲（すなわち、 $1 \leq$ 配列中の要素 ≤ 1 ）に含まれるため、当該要素は、第1行（2行目）の構造体配列に関連していることがわかる。次いで、この構造体配列に属するオフセット配列中のオフセット値が取り出され（ステップ2403）、要素にオフセット値を加えた値が出力される（ステップ2404）。このように、値変換配列を経た値は、出力値配列の格納位置番号に対応する領域に記憶されても良い（ステップ2405）。或いは、値変換配列を経た値を直接他の処理に利用しても良いことは言うまでもない。上記格納位置番号が「0」の要素に関しては、対応するオフセット値は「1」であるため、「 $1+1=2$ 」が変換配列後の要素（格納値）として出力される。実体配列中の必要な値に関して、ステップ2402ないし2405の処理を、繰り返すことにより、図23（a）に示すような論理関係を維持しつつ、実体配列の要素自体を変化させることなく、要素の更新をなすことが可能となる。

【0059】〔値変換配列の生成〕次に、上記値変換配列の作成手順の一例につき、図25のフローチャートを用いて説明を加える。図25の例においては、図23（b）に示すものと同様に、ある数以上の値に所定の数を加えるための値変換配列を生成している。なお、この処理も、図1に示すCPU12により実行される。まず、必要な場合に、初期的な値変換配列が生成される（ステップ2500）。このステップの処理は、値変換配列が生成されていなかった場合にのみ、実行される。図23（b）に示す実体配列2301に関しては、図26（a）に示すような、初期的な値変換配列が得られる。初期的な値変換配列2601においては、終了位置配列中の要素として、実体配列中の要素の最大値が割り当てられ、オフセット配列中の要素として「0」が割り当てられる。

【0060】次いで、あるオフセットを与える要素（格納値）の範囲に基づき、生成すべき配列の行数が決定される（ステップ2501）。たとえば、図23（a）に示すように、「1」以上の値をもつ要素に「1」を加えるという条件が与えられた場合には、終了位置配列およびオフセット配列の行数は「2」となる。これは、「1」より小さな値を持つ要素に関する行（或いは構造体配列）、および、「1」以上の値をもつ要素に関する

行（或いは構造体配列）が生成されるべきだからである。その後、昇順で、各範囲の最大値が終了位置配列の対応する行に配置されるとともに、そのオフセット値が、オフセット配列の対応する行に配置される（ステップ2502、2503）。図23（a）に示すような実体配列において、「1」以上の値を持つ要素に「1」を加えるべき場合には、終了位置配列において、第0行（1行目）に、1より小さい値の最大値である「0」が配置され、オフセット配列の第0行（1行目）に、オフセット値「0」が配置される。すなわち、第0行の構造体配列に数値が与えられる。その一方、終了位置配列において、第1行（2行目）には、実体配列中の要素の最大値「1」が与えられる。これは、値変換配列の初期値を用いれば良い。また、オフセット配列の第1行（2行目）に、オフセット値「1」が配置される（図26（a）の符号2602参照）。上記ステップ2502、2503の処理を条件にしたがった全ての範囲に施すことにより（ステップ2504、2505参照）、値の変換条件にしたがった値変換配列を得ることが可能となる。

【0061】なお、上記実施の形態では、1つの条件により、2つの変換配列の領域が形成されたが、これに限定されるものではなく、複数の条件（たとえば、 $0 \leq \text{値} \leq 1$ である場合には、要素に「1」を加え、 $1 \leq \text{値} \leq 4$ である場合には、要素に「-1」を加え、その他の要素に関してはオフセットを与えない）に基づいて、値変換配列を作成できることは明らかである（図26（b）の符号2611、2612参照）。上記添え字変換配列および値変換配列は、組み合わせて利用することができる。たとえば、図27においては、「0」ないし「9」の要素が繰り返し現れる実体配列（符号2700参照）の上流側（入力側）に、「0」から「500、000」までの添え字にはオフセット「0」を与え、「500、000」から「999、999」までの添え字にはオフセット「1」を与える添え字変換配列（符号2701参照）を設け、実体配列の下流側（出力側）に、値が49以下であればオフセット「-1」を与え、値が50以上であればオフセット「1」を与える値変換配列（符号2702）を設けている。これにより、添え字が、添え字変換配列、実体配列および値変換配列を経て、出力値を得ることができる。

【0062】上記添え字変換配列を用いて、ある情報ブロック中の実体配列に与えられる添え字を変換し、変換済の添え字により実体配列中の要素（項目値）を取り出し、さらに、上記値変換配列を用いて、取り出された要素（項目値）を変換することにより、当該情報ブロックの入力（添え字）と出力（変換済の値）とからなる見かけの（論理的な）配列を考えることが可能となる。

【0063】〔具体的な表形式データに対するデータの削除〕次に、上述した手法を用いて、実際の表形式デー

タの所望のデータを挿入し、削除し或いは更新する具体的な手順につき説明を加える。第5の実施の形態では、会員姓および会員名という項目からなる人名テーブルに、所望の会員姓名が挿入、削除され、或いは、その姓或いは名が更新される。ここでは、挿入に関して第2の実施の形態にかかる手法を用いている。すなわち、添え字変換配列および値変換配列が、それぞれ、終了位置配列およびオフセット配列から構成されるようになっている。なお、以下に述べる添え字変換配列の生成処理、および、値変換配列の生成処理には、新たに変換配列を生成することと、もともと存在した変換配列を更新することとが含まれる。したがって、以下において、「生成」は、生成や更新を意味し、場合によっては、これを「生成（ないし更新）」と称する。

【0064】図28（a）は、人名テーブルの例を示す図である。このような人名テーブル2800のために、本実施の形態においては、図28（b）に示すレコード番号リスト2810、図28（c）に示す「会員姓」という項目に関する情報ブロック2820、および、図28（d）に示す「会員名」という項目に関する情報ブロック2830が生成される。情報ブロック2820の値リストへのポインタ配列2821において、初期的な添え字変換配列2823および初期的な変換配列2824は予め作成されている。また、実体配列2825が、変換配列を作成する前の（もとの）値リストへのポインタ配列に対応する。値リスト2822においても、初期的な添え字変換配列2826が予め作成されている。なお、実体配列2827が、もともとの項目値を格納した値リストに対応している。情報ブロック2830の値リストへのポインタ配列2831においても、初期的な添え字変換配列および値変換配列が予め作成されており、値リスト2832においても、初期的な添え字変換配列が予め作成されている。

【0065】このような構造を有する情報ブロックからなる表形式データにおいて、図29に示すように、3つあるレコードのうち（図29（a）参照）、第1行（2行目）のレコードの削除する（図29（b）参照）場合を考える。この場合には、以下に述べるような各情報ブロックに関して、〔添え字変換配列による要素の削除〕にて述べた処理が実行される。まず、「会員姓」の情報ブロックに関して考察を加える。図29（a）のような「会員姓」を得るために、初期的には、値リストへのポインタ配列において、図30の符号3000に示すような添え字の入出力関係が成立している。すなわち、添え字である「レコード番号」の入力に対して、実体配列である「値リストへのポインタ配列」のポインタ値が、符号3000に示すように出力される。

【0066】さて、このような「値リストへのポインタ配列」中の添え字変換配列に対して、図1に示すCPU12により、図17に示すものと略同様の処理が施され

る。より具体的には、まず、実体配列（ポインタ配列）3001において削除すべき要素の位置等が特定され（図17のステップ1701）、削除位置に関する変換配列の領域が確保される（ステップ1702）。次いで、削除位置の前に関して、終了位置配列の要素として、「削除位置-1」すなわち「0」が配置される（ステップ1703）。なお、この例では、開始位置配列が設けられていないため、ステップ1704およびステップ1705が省略される。その後、削除位置より後に関して、終了位置配列の各要素がデクリメントされる（ステップ1706）とともに、オフセット配列の各要素がインクリメントされる（ステップ1707）。このような処理により、図30（b）に示すような、添え字変換配列を得ることができる。図30（b）において、添え字であるレコード番号に対して、実体配列である「値リストへのポインタ配列」のポインタ値が、符号3010に示すように出力されることが理解できるであろう。

【0067】また、「会員名」に関する情報ブロック（図28の符号2830参照）に関しても、「会員姓」に関する情報ブロックと同様に、CPU12により、図17に示すものと略同様の処理が施される。これにより、図31（a）に示す「会員名」の情報ブロック中の添え字変換配列3101が、図31（b）に示すような添え字変換配列3111になる。したがって、添え字である「レコード番号」の入力と、実体配列である「値リストへのポインタ」のポインタ値との関係は、図31（a）の符号3100に示すものから、図31（b）の符号3110に示すものに変化する。

【0068】このように添え字変換配列の生成処理を施した後に、レコード番号からポインタ配列を経ることにより、値リストの実体配列中において所望の要素（項目値）を特定することができる。図32に示すように、CPU12は、新たなレコード番号（添え字）から、添え字変換配列を経た新たな添え字により、実体配列中の要素（ポインタ値）を特定する。たとえば、レコード番号「0」に関するオフセット値は「0」であるため、実体配列中の第0行（1行目）の要素（ポインタ値）が取り出され、このポインタ値「1」によって、値リスト中の実体配列における第1行（2行目）の項目値が特定される（図32において①参照）。これに対して、レコード番号「1」に関するオフセット値は「1」であるため、「1+1=2」という変換済の添え字が得られ、これにより、実体配列中の第2行（3行目）の要素（ポインタ値）が取り出される。次いで、このポインタ値「1」によって、値リスト中の実体配列における第1行（2行目）の項目値が特定される（図32において②参照）。

【0069】「会員名」の情報ブロックに関しても、同様の経路（③、④参照）により、値リスト中の実体配列の項目値が特定されることは明らかであろう。このよう

にして、新たなレコード番号に対応する人名テーブルの表（ビュー）3200を得ることが可能となる。ビュー3200において、①ないし④は、上記処理において特定された項目値をそれぞれ示す。

【0070】[具体的な表形式データに対するデータの挿入]次に、データを挿入する例につき説明を加える。たとえば、2つあるレコード（図33（a）参照）のうち、第0行（1行目）と第1行（2行目）との間に、あるレコードを挿入する（図33（b）参照）を考える。この場合には、以下に述べるような各情報ブロックに関して、「添え字変換配列による要素の挿入が実行されるとともに、必要な情報ブロックに関して「値変換配列の生成」が実行される。図34は、レコード（データ）の挿入前の各情報ブロックの状態を示す図である。まず、「会員姓」の情報ブロックにおける添え字変換配列の更新、および、「会員名」の情報ブロックにおける添え字変換配列の更新につき考察を加える。

【0071】「会員姓」に関しては、値リストの実体配列中に、「佐藤」という要素（項目値）を加える必要がある。そこで、CPU12は、値リスト中の添え字変換配列による挿入処理（図13参照）を実行する。より具体的には、要素（項目値）の挿入位置が決定される（図13のステップ1300参照）。この例では、「佐藤」という要素（項目値）を挿入する必要があるが、値リストの実体配列においては、要素が昇順（この場合には五十音順）で配置されているため、「佐藤」という要素は、「鈴木」という要素の前に配置する必要があり、したがって、その挿入位置は第0行（1行目）と判断される。そこで、図13のステップ1301にしたがって、各変換配列の領域が確保される。この例では、挿入位置が第0行（1行目）であるため、挿入位置より前には要素が存在せず、挿入位置および挿入位置より後の要素に対して、ステップ1303ないしステップ1307の処理が施される。図35（b）に示すように、挿入位置（符号3510参照）に関して、終了位置配列に、挿入位置を示す要素「0」が配置され（符号3511参照）、かつ、実体配列の末尾を示す位置「1」が、オフセット配列の対応する要素として配置される（符号3512参照）。また、挿入位置より後に関して、終了位置配列に、もとの値に「+1」を加算した値が配置され（符号3513参照）、オフセット配列の対応する値として、もとのオフセット値「0」に「-1」を加算した値が配置される（符号3514参照）。また、実体配列の末尾に、新たな要素（項目値）「佐藤」が配置される（符号3520参照）。実体配列においては挿入すべき新たな要素が末尾に付加された形態となっているが、値リスト中の添え字変換配列により、見かけの（つまり論理的な）配列は、符号3521にて示すように、昇順（五十音順）となっている。

【0072】同様に、「会員名」に関しても、値リスト

の実体配列中に、「エー作」という要素(項目値)を加える必要がある。そこで、CPU12は、「会員名」の情報ブロックにおいて、の値リスト中の添え字変換配列による挿入処理(図13)を実行する。より具体的には、要素(項目値)の挿入位置が、第1行(2行目)であると決定され(図36(b)の符号3610参照)、各変換配列の領域が確保される。挿入位置より前の要素に対しては、その終了配列中の要素として、挿入位置(この場合「1」)から「1」を減じた(「-1」を加算した)値が割り当てられ(符号3611参照)、その一方、対応するオフセット配列中の要素は保持される(符号3612参照)。また、挿入位置(符号3610参照)に関して、終了位置配列に、挿入位置を示す要素「1」が配置され(符号3513参照)、かつ、実体配列の末尾を示す位置「2」を示すためのオフセット値「1」が、オフセット配列中の要素として与えられる(符号3614参照)。

【0073】その一方、挿入位置より後に関して、終了位置配列に、もとの値に「+1」を加算した値が配置され(符号3615参照)、オフセット配列の対応する値として、もとのオフセット値「0」に「-1」を加算した値が配置される(符号3516参照)。さらに、実体配列の末尾に、新たな要素(項目値)「エー作」が配置される(符号3620参照)。なお、実体配列においては挿入すべき新たな要素が末尾に付加された形態となっているが、値リスト中の添え字変換配列により、見かけの(つまり論理的な)配列は、符号3621にて示すように、昇順(五十音順)となっていることが理解できる。このようにして各情報ブロックの値リストに関する、値変換配列の生成(ないし更新)が実現された後、これに付随した、ポインタ配列の側の添え字変換配列および値変換配列の生成(ないし更新)が実行される。

【0074】まず、値リストへのポインタ配列において、実体配列の要素を変換するための値変換配列が生成(ないし更新)される必要がある。たとえば、図37に示す「会員姓」の情報ブロックにおける値リストへのポインタ配列に関して、関連する値リスト(すなわち「会員姓」の値リスト)に、先の処理により、新たな姓「佐藤」が見かけの(つまり論理的な)配列の先頭に挿入されている(図35の符号3521参照)。そこで、値リストへのポインタ配列のポインタ値(すなわち実体配列3701における要素)のうち、0以上の値を有するものがすべてインクリメントされなければならない。そこで、CPU12は、図25に示すフローチャートにしたがって、上記値リストへのポインタ配列における値変換配列を生成(ないし更新)する。

【0075】図37(a)に示す配列では、値のとり得る最大値が「0」であり、かつ「0」以上の値をインクリメントする必要があるため、結果として、オフセット配列中の値「0」に「+1」が加算される(図25のス

テップ2503、図37(b)の符号3711参照)。初期的には、実際に実体配列中に収容された要素(ポインタ値)と、値変換配列を経て出力される要素(ポインタ値)とが、図37(a)の符号3702に示すように同一であったのに対して、値変換配列の生成(ないし更新)の後には、値変換配列を経て出力される要素(ポインタ値)が、図37(b)の符号3712に示すように、もとの値をインクリメントした形態となる。また、新たな姓「佐藤」がレコード番号「1」(すなわち第1行或いは2行目)に挿入されることに伴って、値リストへのポインタ配列における添え字変換配列の生成(ないし更新)処理を実行するとともに、値リスト中の見かけの(つまり論理的な)配列の第0行(1行目)にも姓「佐藤」が挿入されたため、値リストへのポインタ配列中の値を、上記値リストの変更に伴って生成(ないし更新)する処理を実行する必要がある。つまり、値リストへのポインタ配列では、添え字であるレコード番号から整数化されたポインタ値を取り出すものであり、したがって、以下の添え字変換配列の生成(ないし更新)により、添え字変換配列を生成して、レコード番号「1」に対応する姓「佐藤」を示すことができるようにするとともに、ポインタ配列からの出力(値)を変換して、「佐藤」および「鈴木」という要素を昇順で並べた見かけの(つまり論理的な)値リストを適切に示すことができるようにしている。

【0076】まず、CPU12は、上記ポインタ配列の添え字変換配列に関して、添え字であるレコード番号が挿入されたことに対応して、図13に示す処理を実行する。より具体的には、新たな姓「佐藤」(および新たな名「エー作」)を挿入すべきレコード番号が「1」であるため、挿入位置が第1行(2行目)であると判断され(図13のステップ1301参照)、挿入位置およびその前後に関する変換配列の領域が確保される(ステップ1302)。以下、ステップ1303ないしステップ1308の処理(実際には、開始位置に関する処理が省略される)により、図38(a)に示すように添え字変換配列(符号3810参照)が形成される。

【0077】これにより、添え字変換配列を経ることにより、値リストのポインタ配列は、添え字であるレコード番号の入力に対して、符号3811に示すような出力が選られることになる。ところで、図37を参照して説明したように、上記例では、値リストにおける見かけの(つまり論理的な)配列では、昇順(五十音順)にしたがって、「佐藤」という要素(項目値)が「鈴木」という要素(項目値)より上位に位置していることがわかっていて、したがって、値変換配列により、値リストのポインタ配列中の実体配列の値を変換して、値リスト中の要素(項目値)を適切に指示できるようにしなければならない。そこで、CPU12が図25にしたがった処理を実行する。この処理では、姓「佐藤」の挿入により、

実体配列の値が「1」であるものの出力が「0」となるようにオフセット値、すなわち、オフセット値「-1」を与え、それ以外についてはオフセット値「1」を与えるような処理を実行する。これにより、図38(b)に示すように値変換配列(符号3820参照)を得ることができる。

【0078】図38(b)において、新たなレコード番号(添え字)に対して、添え字変換配列、実体配列および値変換配列を経て出力された値は、見かけの(つまり論理的な)配列(符号3821参照)に示すようになり、これにより、適切に値リストの要素(項目値)が特定されることが理解できよう。「会員姓」の情報ブロックの値リストへのポインタ配列に対して施されたものと同様の処理が、「会員名」の情報ブロックに対しても施される。まず、図39に示すように、「会員名」の情報ブロックのうち、値リストに関する処理(添え字変換配列の生成(更新)および実体配列への要素(項目値)の挿入:図36参照)に伴う値リストへのポインタ配列の生成(ないし更新)が実行される。CPU12が図13に示す処理を事項することにより、まず、図39(b)に示す値変換配列(符号3910参照)を得ることができるとは明らかであろう。

【0079】次いで、CPU12は、上記ポインタ配列の添え字変換配列に関して、添え字であるレコード番号が挿入されたことに対応して、図13に示す処理を実行する。より具体的には、新たな会員名「エー作」が挿入すべきレコード番号が「1」であるため、第1行(2行目)を挿入位置として(図13のステップ1301参照)、図13にしたがった処理が行われる。これにより、図40(a)に示すようなオフセット配列(符号4010参照)を得ることができる。さらに、上記添え字変換配列の生成(ないし更新)および実体配列の末尾への要素の挿入に伴って値変換配列を更新する必要があるため、CPU12は、図25に示す処理を実行し、これにより、図40(b)に示す値変換配列(符号4020参照)を得ることができる。

【0080】図40(b)において、レコード番号を添え字とする値リストのポインタ配列の出力は、見かけの(論理的な)配列(符号4021参照)に示すものとなる。各レコード番号からは図40(b)の破線で示すような経路で出力が得られていることが理解できよう。このように、各情報ブロックにおいて添え字変換配列および/または値変換配列の生成ないし更新処理を施した後、レコード番号からポインタ配列を経ることにより、値リストの実体配列中において所望の要素(項目値)を特定することができる。図40に示すように、CPU12は、値リストへのポインタ配列において、新たなレコード番号(添え字)から、添え字変換配列、実体配列および値変換配列を経て、値リストの実体配列中の要素(項目値)を特定するための出力を得る。たとえば、

「会員姓」の情報ブロックに関して、レコード番号「0」に関するオフセット値は「0」であるため、実体配列中の第0行(1行目)の要素(ポインタ値)が取り出され、さらに、この要素(ポインタ値)の対応するオフセット値が「1」であるため、値リストへのポインタ配列からの出力は「1」となる(図40において①参照)。これに対して、レコード番号「1」に関して、添え字変換配列のオフセット値は「1」であるため、「1+1=2」という変換済の添え字が得られ、これにより、実体配列中の第2行(3行目)の要素(ポインタ値)が取り出される。さらに、この要素(ポインタ値)に対応するオフセット値が「-1」であるため、値リストへのポインタ配列からの出力は「0」となる(図40において②参照)。同様にしてレコード番号「3」についても、出力「1」を得ることができる。このような出力は、値リストに与えられ、値リスト中の添え字変換配列を経たものにより、実体配列中の要素(項目値)が特定される。「会員名」の情報ブロックに関しても、同様の経路を経て、レコード番号から値リストの実体配列中の要素(項目値)が特定される(図41の④ないし⑥参照)。

【0081】このようにして、新たなレコード番号に対応する人名テーブルの表(ビュー)4100を得ることが可能となる。ビュー4100において、①ないし⑥は、上記処理において特定された項目値をそれぞれ示す。本実施の形態によれば、実体配列への入力を添え字として定義し、当該添え字を変換し、および/または、実体配列からの出力である値を変換することにより、実体配列の要素自体を移動することなく、実体配列への要素の挿入および削除を実現している。つまり、実体配列への入力および出力に所定の関数を施すことにより、仮想的に(或いは論理的に)実体配列に対する挿入/削除/更新が行われたとみなせるように構成している。したがって、実体配列中の要素の移動を必要とすることなく、データの挿入等を実現することができる。特に、要素数が膨大な表形式データにおいて、要素の移動を伴わないことにより、処理の著しい高速化に寄与することが可能となる。

【0082】[表形式データに対するデータの更新]上述したように、本発明においては実体配列の変更(すなわち、データ自体の変更や移動)を伴うことなく、表形式データに対するデータの挿入および削除を実現することができる。表形式データに対するデータの更新は、あるレコード番号のデータの削除(レコードの削除)および同じレコード番号へのデータの挿入(レコードの挿入)を順次実行することと等価である。たとえば、図28に示す表形式データ(人名テーブル)において、第1行(2行目)のレコードを更新する行為(たとえば、「会員姓」の「佐藤」を「田中」に更新する処理、および/または、「会員名」の「エー作」を「ビー作」に更

新する処理)を考える。この場合に、まず、図29ないし図32を参照して説明した手法を用いて、第1行(2行目)のレコードを削除する処理を実行し、その後、図33ないし図42を参照して説明した手法を用いて、第0行(1行目)と第1行(2行目)との間に、新たなレコードを挿入する処理を実行すれば良い。

【0083】[情報ブロックの形態にて表わされた表形式データの挿入等に関する考察]さて、上記情報ブロックの形態にて表わされた表形式データにおいて、データを追加する場合につき、再度考察を加える。前述したように、上述した例では、図46(a)に示すように、CPU12により、値リストへの項目値の挿入に伴う添え字変換配列の処理、すなわち、五十音順にソートされた状態にて項目値が収容された状態の「見かけの(論理的な)配列」を得るために添え字変換配列を生成(ないし更新)する処理が実行される(ステップ4601)。その後、上記処理において、見かけの(論理的な)実体配列(値リスト)の所定の位置に、要素(項目値)が挿入されているため、これに伴って、値リストへのポインタ配列に関して、所定の位置に対応する値以上の値を有するものがインクリメントされている(ステップ4602)。次いで、レコード番号とポインタ配列中のポインタ値に着目して、CPU12は、レコード番号の挿入に伴って、値リストへのポインタ配列の添え字変換配列の処理を実行している(ステップ4603)。このような処理の後、最後に、新たに挿入されたポインタ配列の実体配列における項目値(前記例では、もとの実体配列中の最大値よりも一つ大きな値)に対する値変換配列を生成している(ステップ4604)。

【0084】この処理を見ると、ステップ4601とステップ4603とは独立して実行でき(場合によっては並列的に実行することもでき)、その一方、ステップ4602とステップ4604とは一括して実行することができる。したがって、挿入処理は、図46(b)に示すように、ポインタ配列の添え字変換配列に関する処理(ステップ4611:図46(a)のステップ4603に対応する)、値リストの添え字変換配列に関する処理(ステップ4612:ステップ4601に対応する)、および、ポインタ配列の値変換に関する処理(ステップ4613:ステップ4602および4604に対応する)からなると考えることができる。さて、ここで、 m 個のレコードに対応して m 個の要素(ポインタ値)からなる値リストへのポインタ配列と、 n 個の要素(項目値)からなる値リストとを考える。ここで、第 j 行のレコードに、要素を挿入すると仮定する。図47(a)に示すように、第 j 行の要素の挿入に対して、ステップ4611の処理により、値リストへのポインタ配列に関して、終了位置配列およびオフセット配列からなる添え字変換配列(符号4701参照)、および、その末尾に新たなポインタ値が格納された実体配列(符号4702参

照)が得られることが理解できるであろう。

【0085】その一方、所定の順序(たとえば五十音順)にて要素(項目値)が配置された値リストに関して、この要素を走査することにより、第 i 行に要素を挿入すべきであると判断された場合にも、ステップ4612の処理により、終了位置配列およびオフセット配列からなる添え字変換配列(符号4703参照)、および、その末尾に挿入すべき要素(項目値)が配置された実体配列(符号4704参照)が得られることが理解できるであろう。ここで、値リストへのポインタ配列において、添え字変換配列は、添え字(レコード番号)「 p 」が、 $0 \leq p \leq (j-1)$ である場合には、オフセット値「0」を加え、 $j \leq p \leq j$ (すなわち $p=j$)の場合には、オフセット値「 $m-j$ 」を加え、或いは、 $(j+1) \leq p \leq m$ の場合には、オフセット値「-1」を加えることを意味している。また、値リストにおいて、添え字変換配列は、入力した添え字(ポインタ配列の出力)

「 q 」が、 $0 \leq q \leq (i-1)$ である場合には、オフセット値「0」を加え、 $i \leq q \leq i$ (つまり $q=i$)の場合には、オフセット値「 $n-i$ 」を加え、或いは、 $(i+1) \leq q \leq n$ の場合には、オフセット値「-1」を加えることを意味している。

【0086】次に、値リストへのポインタ配列における値変換配列に考察を加えると、値リストの見かけ(論理上)の挿入位置「 i 」より小さな値についてはオフセットを加える必要が無い(すなわち、ポインタ値の意味合いに変化が無い)と考えることができる。その一方、上記挿入位置「 i 」以上の値については、見かけ(論理上)の値リストに対してポイントすべき位置が、一つずつ下方(すなわち値の大きな方)にシフトしていることになるため、その値にはオフセット値「1」を加えるべきであることが理解できる。さらに、値リストの実体配列の末尾に追加されたポイント値(すなわち、レコード番号の挿入処理に伴って付加された、実体配列中の位置「 m 」におけるポインタ値)の値は、「 n 」であると考えられる。これは、挿入処理前のポインタ配列の実体配列において、「0」ないし「 $n-1$ 」の何れかの値を取り得るため、挿入処理によりポイント値「 n 」が与えられるからである。このため、このポイント値「 n 」が値変換により、値リストの見かけ(論理上)の挿入位置「 i 」をポイントするためには、そのオフセット値を「 $i-n$ 」に設定すればよいことがわかる。無論、実体配列中の位置「 m 」におけるポイント値が他の値を割り当てても、値変換により、その値が「 i 」に変換されるようなオフセット値を与えればよい。このようにして、図47の符号4705に示すような値変換配列を得ることが可能となる。すなわち、ポインタ配列の値変換リストに関して、実体配列中の要素(ポインタ値)「 r 」が、 $0 \leq r \leq (i-1)$ である場合には、オフセット値0を加え、 $i \leq r \leq (n-1)$ の場合には、オフセット値

「+1」を加え、或いは、 $n \leq r \leq n$ (つまり $r = n$) の場合には、オフセット値「 $i - n$ 」を加えればよいことになる。

【0087】削除に関しても、挿入の場合と同様に、 m 個のレコードに対応して m 個の要素 (ポインタ値) からなる値リストへのポインタ配列と、 n 個の要素 (項目値) からなる値リストを考え、その第 j 行のレコードを削除すると仮定する。この場合には、図48に示すように、ポインタ配列中の添え字変換配列 (符号4801) 参照を得ればよい。図48から理解できるように、添え字 (レコード番号) 「 p 」が、 $0 \leq p \leq (j - 1)$ の場合には、オフセット値0を加え、 $j \leq p \leq (m - 2)$ の場合には、オフセット値「+1」を加えればよい。これにより、変換後の添え字は、もとの (変換前の) 添え字「 j 」に対応するポインタ値を、ポインタすることがなくなり、これにより当該ポインタ値およびこれにより特定される値リストが、見かけ上 (論理的に) 削除される。

【0088】[他の手法による表形式データに対するデータの削除] 次に、本発明において、表形式データに対するデータの削除の他の例につき説明を加える。この手法は、データの挿入がなされない場合に有効である。再度、図28のような構造を有する情報ブロックからなる表形式データにおいて、図29に示すように第1行 (2行目) のレコードを削除する場合を考える。

【0089】本例では、「会員姓」および「会員名」の情報ブロックに処理を施すことなく、レコード番号リストに処理を施すことにより、レコードの削除を実現している。より詳細には、図42に示すように、レコード番号リストの添え字変換配列に関して、削除の処理を実行し、もとのレコード番号「1」を削除した形態の論理的な出力をなすようにしている。図42(a)に示すように、レコード番号リストにおける添え字入力とレコード番号出力との関係は、符号4201に示すようなものとなっている。そこで、CPU12は、第1行 (2行目) を削除するために、図17に示す処理を実行する。より詳細には、削除位置 (この場合には第1行) が特定され (ステップ1701)、次いで、削除位置に関する変換配列の領域が生成され (ステップ1702)、変換配列中の要素につき、所定の処理が実行される (ステップ1703ないし1707参照、ただし、この例では、開始配列に関する処理は実行されていない)。これにより、図42(b)に示すような添え字変換配列 (符号4210参照) が得られる。この添え字変換配列を経ることにより、入力と出力の関係は符号4211に示すようなものとなり、論理的にはレコード番号「1」が削除され、それ以下のレコード番号が一つずつシフトした状態が実現されていることが理解できるであろう。本例によれば、レコード番号自体の添え字変換配列を用いることにより、仮想的 (論理的) にレコード番号の削除および

シフトが実現され、したがって、極めて簡素な手法によりレコードの削除を実現することが可能となる。

【0090】[本発明のトランザクション処理等への応用] 次に、上記手法を用いてトランザクションを適切になすことにつき説明を加える。上述したように、本発明においては実体配列の要素の順序を変更したり、或いは、要素を移動することなく、要素の挿入、削除並びに更新を実現している。そこで、これを用いれば、トランザクションを適切に実現することができる。図43は、トランザクション処理を示すフローチャートである。この処理は、図1のCPU12により実行される。また、表形式データを複数の情報ブロックの集合体の形態とするための処理 (たとえば、図6に示す処理) は予め実行されているものとする。

【0091】CPU12は、まず、トランザクション処理に関わる全ての情報ブロックに関して初期的な変換配列を付与する (ステップ4301)。たとえば、レコード番号リストにおいては、レコード番号の配列 (実体配列) に対する初期的な添え字変換配列を生成し、或いは、ある情報ブロックの値リストへのポインタ配列においては、ポインタ配列の実体配列に対する初期的な添え字変換配列および値変換配列を生成するとともに、当該情報ブロックの値リストにおいては、実体配列 (実際の値リスト) に対する初期的な添え字変換配列を生成する。次いで、CPU12は、所定のトランザクション処理を実行する (ステップ4302)。ここにおいて、レコードの挿入、削除或いは更新の何れかが必要である場合には、前述したように、添え字変換配列や値変換配列を更新し、かつ、必要な場合には実体配列の末尾に要素を挿入 (或いは別の領域に差分配列を生成) する。たとえば、オペレータ等によりロールバックの指示があった場合 (ステップ4303でイエス(Yes)には、CPU12は、その時存在した全ての変換配列を廃棄する (ステップ4304)。

【0092】その一方、ロールバックの指示がなく (ステップ4303でノー(No))、かつ、トランザクション処理が終了した場合には、CPU12は処理をコミットする (ステップ4304)。コミットの際に、CPU12は各変換配列を破棄しても良いし、或いは、各変換配列の状態を、ステップ4301にて生成した初期的な変換配列のものに戻し、他のプロセスがこの状態の変換配列を経るように構成しても良い。また、処理終了時の変換配列の状態のまま、他のプロセスがこれを利用して良い。さらに、変換配列を維持して、必要に応じてこれらを利用できるようにしても良い。

【0093】次に、同じ表形式データを用いた複数のプロセスが並列的に実行される場合を考える。たとえば、あるプロセスがデータの更新を伴っており (たとえば、口座から1万円を引き出す処理)、その一方、他のプロセスが表形式データを参照している (たとえば、各口座

の預金残高の総領を求める処理)場合を考える。上記データの更新を伴うプロセス(「更新プロセス」と称する)においては、コミットが生じるまで、1万円が仮に引き出された状態となる。本発明においては、更新プロセスが、添え字変換配列や値変換配列を用いてレコードの削除および挿入を行っており(図44の実線の矢印参照)、その一方、他のプロセス(「参照プロセス」と称する)は、上記添え字変換配列や値変換配列を参照することなしに、直接実体配列を参照する(図44の破線の矢印参照)ことにより、複数の処理が並列的に実行されても、それぞれが干渉することなく適切に処理をなすことができる。

【0094】次に、レコードのロックの手法につき簡単に説明を加える。この場合には、各レコード番号に対応つけて、ロックを管理する情報ブロックを設ければ良い。図45は、レコードのロックを説明するための図である。図45に示すように、表形式データにおいて、ロック管理の情報ブロック(符号4501)が設けられ、当該情報ブロック中に、レコード番号数の要素を配置できるポインタ配列(符号4502)を設けている。このポインタ配列には、要素(格納値)として、読み書き不可能を示す「NON」、読み出し専用(更新や削除の不可)を示す「R」、読み書き可能を示す「RW」などが格納されている。無論、このような文字を用いる必要なく、それぞれのステータスを示す数値(たとえば、読み書き不可能であれば「0」、読み出し専用であれば「1」など)を割り当てれば良いことは言うまでもない。このように、レコード番号に対応したポインタ配列を参照することにより、プロセスは当該レコード番号に対応する種々のデータの読み書き可能性を知ることができるとともに、プロセスが、処理の際に、所定のレコード番号に対応するポインタ配列を更新することにより、排他処理を実現することが可能となる。

【0095】このように、各レコードに対応してロックの属性(種別)を含む項目値を配置した配列を設けることにより、まず、大量のレコードに対して、高速にロックをなすことが可能となる。たとえば、ある項目に関して特定の項目値を有するレコード(たとえば、性別という項目で「女性」という項目値を有するレコード)を、高速にロックすることができる。すなわち、従来では、OSコールにてロックしたいレコードを一つ一つOSに登録すべきところ、本発明によれば、情報ブロックの配列中の項目値を変更することで、OSコールを介することなく、そのロックをなすことが可能となる。また、上記構成によれば、表形式データをジョインさせた場合にも、そのロックを伝播することが可能となる。たとえば、特願平11-151156号にしたがって、各々が複数の情報ブロックの集合体である複数の表形式データのうち、共通の意味合いを有する情報ブロックの項目値を共通化することにより、ジョインを実現する場合を考

える。まず、各表形式データに関して、ロックに関する情報を格納する配列を含む情報ブロックが生成される。ここに、ジョインの処理をなす前の各表形式データの情報ブロック中の配列を原テーブルと称する。次いで、ジョインの処理がなされるが、この場合、原テーブルのポインタ配列は、基本的に保存されるため、ジョインにより生成されたテーブルから、原テーブルのロック情報を格納する配列を含む情報ブロックを参照ないし書き換えることが可能である。

10 【0096】したがって、ジョインテーブルを作成してこれを表示しながら、ユーザからのデータ変更以来を受け付けることが可能となる。より具体的に、以下にその手順を説明する。

(1) まず、ジョインテーブルを作成し、これが表示される。

(2) 次いで、ユーザがあるジョインテーブル中のレコードを変更することを指示すると、プログラムは、ユーザが指示したレコードにロックがかかっていないことを確認する。このときに、ジョインにより生成されたテーブルには、ロック情報を管理する情報ブロックが存在しないため、実際には、原テーブルにおける該当レコードに関するロック情報を調べる。

(3) 次いで、プログラムは、ジョインテーブル中のそのレコードをロックする。実際には、原テーブルの対応するレコードをロックすることになる。

(4) その後、プログラムは、ユーザの指示にしたがってデータの書き換えを実行する。

(5) 最後に、プログラムは、変更をコミットし、ロックを解除する。

30 このように、上記構成に従えば、ロックをジョインテーブルにも伝播できるため、ジョインテーブルを書き換え可能なものとすることが可能となる。

【0097】本発明は、以上の実施の形態に限定されることなく、特許請求の範囲に記載された発明の範囲内で、種々の変更が可能であり、それらも本発明の範囲内に包含されるものであることは言うまでもない。たとえば、本発明は、前記第5ないし第7の実施の形態のように、複数の情報ブロックの組により構築された表形式データに対するデータの挿入、削除および更新に限定されるものではない。図49に示すように、レコード番号(符号4901参照)と、姓(4文字)、名(4文字)、住所(20文字)等の文字列からなる構造体(符号4902参照)とが対応付けられた形態の表形式データに関するデータの挿入、削除および更新をなすこともできる。たとえば、挿入の場合には、氏名、住所等ならなる構造体配列の組を実体配列と考え、レコード番号を添え字と考えて、図13に示すものと略同等の処理を実行すれば良い。また、削除の場合にも、同様に、構造体配列の組を実体配列と考え、レコード番号を添え字と良

い。

【0098】また、本発明にかかる値変換配列を単独で利用することも可能である。たとえば、D/A変換器（図示せず）から、順次1億サンプルの測定データが与えられる場合を考える。通常、1億サンプルのデータを一つのメモリにて収容することは容易でなく（たとえば、2バイト/1サンプルであっても、2億バイト（100Mバイト以上）のデータ容量を要する）、これを10個のメモリに分割して記憶することを考える。図50において、先頭（第0番）のメモリブロック5000から最後尾（第9番）のメモリブロック5009に、それぞれ1000万個の測定データ（サンプルデータ）が収容されている。

【0099】このような場合には、各測定データの累算値を求め、これを記憶すべき場合を考える。本発明に関して、これは以下の手順により実行することができる。各メモリブロックに関して、その先頭から末尾までの測定データのデータ値を加算する。これは、複数のCPU（この場合には10個が望ましい）が並列的に、自己が演算をなすべきメモリブロックに対して、累算処理を実行すれば良い。これにより、図50の符号5010、5011、…、5019に示すような各ブロックごとの累算値を算出することができる。

【0100】次いで、メモリブロックの上位に隣接するメモリブロック中の最終アドレスに対応する論理上の累算値が取り出されて、これが当該メモリブロックの値変換配列におけるオフセット配列中のオフセット値となる。この例では、一つのメモリブロックにおいて、同一のオフセット値が用いられるため、終了位置配列など他の配列を作成する必要はない。たとえば、メモリブロック5021に関して、そのオフセット値は、 $Z(9, 999, 999)$ （ここで、 z は、論理上の累算値）となり、たとえば、当該メモリブロック5021の先頭アドレスに対応する累算値 $Y(10, 000, 000)$ に、オフセット値 $Z(9, 999, 999)$ を加算することにより、論理上の累算値 $Z(10, 000, 000)$ を求めることができる。また、当該メモリブロック5021の末尾アドレスに対応する累算値 $Y(19, 999, 999)$ に、オフセット値 $Z(9, 999, 999)$ を加算することにより、論理上の累算値 $Z(19, 999, 999)$ が求められる。この累算値 $Z(19, 999, 999)$ は、下位に隣接するメモリブロックにて用いるオフセット値となることが理解できるであろう。

【0101】ここで、累算に要する時間につき考察を加えると、上述したように10個のメモリブロックに対して、並列的な累算処理がなされる場合には、各メモリブロックの累算値を求めるために1000万ステップを要し、また、加算オフセット値の設定のために、9ステップ程度を要する。これに対して、従来の手法で累算値を求めるためには、1億ステップが必要であることが理解

できよう。これは、たとえ、メモリブロックに分割がなされていても、上位に位置するメモリブロックの累算値が算出されて、はじめて当該メモリブロックに関する累算が開始できるため、各メモリブロックに関する並列処理が事実上不可能であることに起因する。このように、本発明の値変換配列を用いることにより、科学技術計算等の高速化も可能となる。

【0102】また、本発明によれば、図51に示すように、ネスト構造をとることも可能である。図51において、最も内側に位置するネストにおいては、添え字変換配列（符号5101参照）および実体配列（符号5102参照）および値変換配列（符号5103）が設けられている。その一方、その外側に位置するネストにおいては、上記添え字変換配列、実体配列および値変換配列からなる組を、第2の実体配列（符号5112参照）と考え、その入力側に添え字変換配列（符号5111参照）を配置するとともに、その出力側に値変換配列（符号5113参照）を配置している。同様に、さらにその外側のネストにおいては、内側のネストを第3の実体配列（符号5212参照）と考え、その入力側に添え字変換配列（符号5211参照）を配置するとともに、その出力側に値変換配列（符号5213参照）を配置している。このような構造をとることにより、階層構造の挿入、削除、更新の処理をなすことが可能となる。たとえば、値の挿入を複数回実行して、その後に更新トランザクションとしてコミットする場合に、上記構造は有用である。

【0103】また、前記図47に示す例では、値リストのポインタ配列において、実体配列の末尾にポインタ値を配置し、そのポインタ値 $f(m) = n$ （ここで、 n は、今までのポインタ値がとり得る最大値「 $n-1$ 」よりも1つ大きい。）と設定し、かつ、値リストにおいても、実体配列の末尾に項目値を配置しているが、このような構成に限定されるものではない。すなわち、値リストへのポインタ配列に関して、実体配列に隣接せずに、他の領域（差分配列）にポインタ値を配置してもよく、或いは、ポインタ値 $f(m)$ も、他の値「 y 」（ただし $y \geq n$ ）と設定してもよい。さらに、値リストに関しても、実体配列に隣接せずに、他の領域（差分配列）に項目値を配置してもよい。図52は、ポインタ配列に関して、領域「 x 」にポインタ値（ $f(m) = y$ ）を配置し、かつ、値リストに関して、領域「 z 」に項目値を配置した例を示している。この場合にも、図47にて説明した場合と略同様に、添字変換配列や値変換配列を求めることが可能である。

【0104】さらに、前記実施の形態において、情報ブロックの形態の表形式データのデータ更新処理は、挿入処理および削除処理、或いは、削除処理および更新処理を順次実行することができることを述べた。このような更新処理に関しても、以下のように、添え字変換配列お

よび値変換配列を求めることができる。たとえば、 m 個のレコード番号と、 n 個の項目値を有する表形式データに関して、レコード番号「 j 」に関して項目値の更新をすべき場合を考える。ここに、更新すべき項目値が挿入された場合の挿入位置（論理上の挿入位置）が「 i 」であるとする。

【0105】まず、値リストのポインタ配列において、実体配列の末尾にポインタ値を配置し、そのポインタ値 $f(m) = n$ （ここで、 n は、今までのポインタ値がとり得る最大値「 $n-1$ 」よりも1つ大きい。）と設定し、かつ、値リストにおいても、実体配列の末尾に更新に利用される項目値を配置する場合につき説明を加える。この場合には、図53に示すように、ポインタ配列の添え字「 p 」に関して、 $0 \leq p \leq (j-1)$ である場合には、オフセット値「0」が与えられ、 $p = j$ である場合には、オフセット値「 $m-j$ 」が与えられ、或いは、 $j+1 \leq p \leq m-1$ である場合には、オフセット値「0」が与えられる。また、ポインタ配列の値変換配列、および、値リストの添え字変換配列は、図47に示すものと同じであることが分かる（削除処理においては、これらに手加えられないため、挿入処理のものが維持される）。

【0106】或いは、更新処理において、ポインタ配列に関して、領域「 x 」にポインタ値（ $f(m) = y$ ）を配置し、かつ、値リストに関して、領域「 z 」に更新される項目値を配置した場合にも、図52に関して説明したものと同様に、添え字変換配列および値変換配列が求められることが理解できよう（図54参照）。また、前記実施の形態においては、一般のコンピュータシステム10内に、所定のプログラムを読み込み、当該プログラムを実行することにより、添え字変換配列や値変換配列の生成、更新等の処理を実現しているが、本発明はこれに限定されるものではなく、パーソナルコンピュータ等のような一般のコンピュータシステムに、データベース処理専用のボードコンピュータを接続し、当該ボードコンピュータが上記処理を実行できるように構成しても良いことは言うまでもない。したがって、本明細書において、手段とは必ずしも物理的手段を意味するものではなく、各手段の機能が、ソフトウェアによって実現される場合も包含する。さらに、一つの手段の機能が、二つ以上の物理的手段により実現されても、若しくは、二つ以上の手段の機能が、一つの物理的手段により実現されてもよい。

【0107】

【発明の効果】本発明によれば、特に、上記線形フィルタ法を用いて、データの挿入、削除および更新の処理を高速かつ適切になすことができる表形式データにおけるデータ挿入、削除および更新方法を提供すること可能となる。本発明は、上記トランザクション処理を適切になすことができる表形式データの挿入、削除および更新

方法を提供することが可能となる。

【図面の簡単な説明】

【図1】 図1は、本発明の実施の形態にかかる検索、集計およびサーチ方法を実現できるコンピュータシステムのハードウェア構成を示すブロックダイアグラムである。

【図2】 図2は、本実施の形態にて用いる情報ブロックを示す図である。

【図3】 図3は、表形式データの例、および、当該表形式データに基づく情報ブロックの例を示す図である。

【図4】 図4は、表形式データの他の例、および、当該表形式データに基づく情報ブロックの他の例を示す図である。

【図5】 図5は、単一項目に関する検索手法を示すフローチャートである。

【図6】 図6は、表形式データに基づき情報ブロックを作成するための処理を説明するフローチャートである。

【図7】 図7は、情報ブロックを作成するための原データの例を示す図である。

【図8】 図8は、配列に要素を挿入する例および配列の要素を更新する例を示す図である。

【図9】 図9は、添え字、配列および配列からの値の関係、および、本発明に係る添え字変換および値変換の概要を説明する図である。

【図10】 図10は、要素の挿入に関する入出力の論理的関係と、第1の実施の形態にかかる配列への要素の挿入を説明するための図である。

【図11】 図11は、第1の実施の形態にかかる添え字から配列中の要素を特定するための処理を示すフローチャートである。

【図12】 図12は、第1の実施の形態において、ある要素の特定の位置への挿入を説明する図である。

【図13】 図13は、本実施の形態において要素の挿入のための添え字変換配列を生成するための処理を示すフローチャートである。

【図14】 図14は、図13の処理により生成された添え字変換配列の一例を示す図である。

【図15】 図15は、要素の削除に関する入出力の論理的関係と、第1の実施の形態にかかる配列への要素の削除を説明するための図である。

【図16】 図16は、第1の実施の形態において、特定の位置の要素の削除を説明する図である。

【図17】 図17は、本実施の形態において要素の削除のための添え字変換配列を生成するための処理を示すフローチャートである。

【図18】 図18は、図17に示す処理にしたがって変換配列中の数値が変化する状態を示す図である。

【図19】 図19は、第1の実施の形態において、挿入および削除を順次実行した場合の状態を説明するため

の図である。

【図20】 図20は、第1の実施の形態において、挿入および削除を順次実行した場合の状態を説明するための図である。

【図21】 図21は、本発明の第2の実施の形態にかかる変換配列の構成を説明するための図である。

【図22】 図22は、本発明の第3の実施の形態にかかる実体配列および差分配列を説明するための図である。

【図23】 図23は、本発明の第4の実施の形態にかかる値変換配列の概略を説明するための図である。 10

【図24】 図24は、第4の実施の形態にかかる値変換の処理を示すフローチャートである。

【図25】 図25は、第4の実施の形態にかかる値変換テーブルを生成する処理を示すフローチャートである。

【図26】 図26は、第4の実施の形態において、実体配列中の値の変換を実行した場合の状態を説明するための図である。

【図27】 図27は、本発明において、添え字変換配列および値変換配列を組み合わせたデータ構造の一例を示す図である。 20

【図28】 図28は、本発明の第5の実施の形態にかかる各情報ブロックの構造を示す図である。

【図29】 図29は、第5の実施の形態において、特定のレコードを削除する処理を説明するための図である。

【図30】 図30は、第5の実施の形態において、特定のレコードを削除する処理を説明するための図である。 30

【図31】 図31は、第5の実施の形態において、特定のレコードを削除する処理を説明するための図である。

【図32】 図32は、第5の実施の形態において、特定のレコードを削除する処理を説明するための図である。

【図33】 図33は、第5の実施の形態において、特定のレコードを特定の位置に挿入する処理を説明するための図である。

【図34】 図34は、第5の実施の形態において、特定のレコードを特定の位置に挿入する処理を説明するための図である。 40

【図35】 図35は、第5の実施の形態において、特定のレコードを特定の位置に挿入する処理を説明するための図である。

【図36】 図36は、第5の実施の形態において、特定のレコードを特定の位置に挿入する処理を説明するための図である。

【図37】 図37は、第5の実施の形態において、特定のレコードを特定の位置に挿入する処理を説明するた 50

めの図である。

【図38】 図38は、第5の実施の形態において、特定のレコードを特定の位置に挿入する処理を説明するための図である。

【図39】 図39は、第5の実施の形態において、特定のレコードを特定の位置に挿入する処理を説明するための図である。

【図40】 図40は、第5の実施の形態において、特定のレコードを特定の位置に挿入する処理を説明するための図である。

【図41】 図41は、第5の実施の形態において、特定のレコードを特定の位置に挿入する処理を説明するための図である。

【図42】 図42は、本発明の第6の実施の形態において、特定のレコードを削除する処理を説明するための図である。

【図43】 図43は、本発明を利用したトランザクション処理を示すフローチャートである。

【図44】 図44は、本発明を利用した複数プロセスの並列処理を説明するための図である。

【図45】 図45は、本発明を利用した排他処理を説明するための図である。

【図46】 図46は、本発明において、情報ブロックの形態をなす表形式データのデータ挿入処理を説明するためのフローチャートである。

【図47】 図47は、本発明において、情報ブロックの形態の表形式データのデータ挿入処理を説明するための図である。

【図48】 図48は、本発明において、情報ブロックの形態の表形式データのデータ削除処理を説明するための図である。 30

【図49】 図49は、本発明にかかる添え字変換配列の他の応用例を説明するための図である。

【図50】 図50は、本発明にかかる値変換配列のさらに他の応用例を説明するための図である。

【図51】 図51は、本発明にかかる添え字変換配列および値変換配列のさらに他の応用例を説明するための図である。

【図52】 図52は、本発明にかかる情報ブロックの形態の表形式データにおいて、データ挿入処理の他の例を説明するための図である。 40

【図53】 図53は、本発明にかかる情報ブロックの形態の表形式データにおいて、データ更新処理を説明するための図である。

【図54】 図54は、本発明にかかる情報ブロックの形態の表形式データにおいて、データ更新処理の他の例を説明するための図である。

【符号の説明】

10 コンピュータシステム
12 CPU

14 RAM
16 ROM
18 固定記憶装置
20 CD-ROMドライバ

* 22 I/F
24 入力装置
26 表示装置
*

【図1】

【図2】

図1

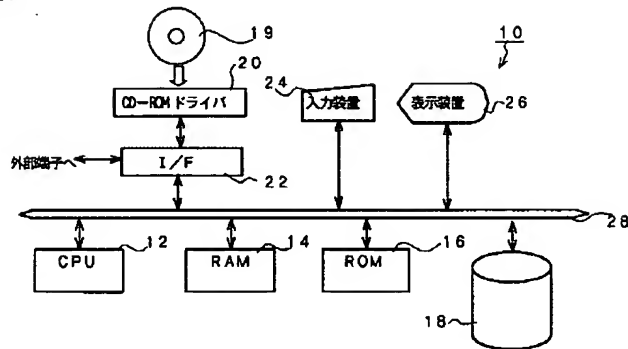
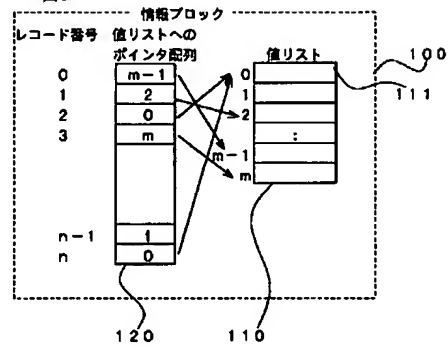


図2

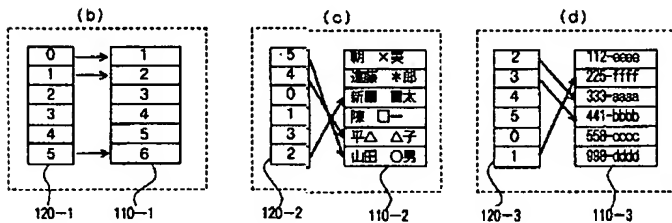


【図3】

【図5】

図3

レコード番号	顧客ID	顧客名	電話番号
0	1	山田 〇男	33-aaa
1	2	平△ △子	41-bbbb
2	3	朝 × 美	58-cccc
3	4	遠藤 × 郎	98-dddd
4	5	陳 □ 一	12-eeee
5	6	新藤 麗太	25-ffff



【図12】

図12

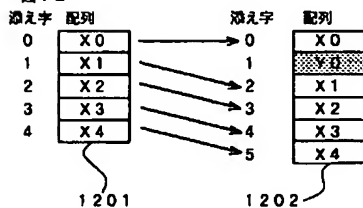
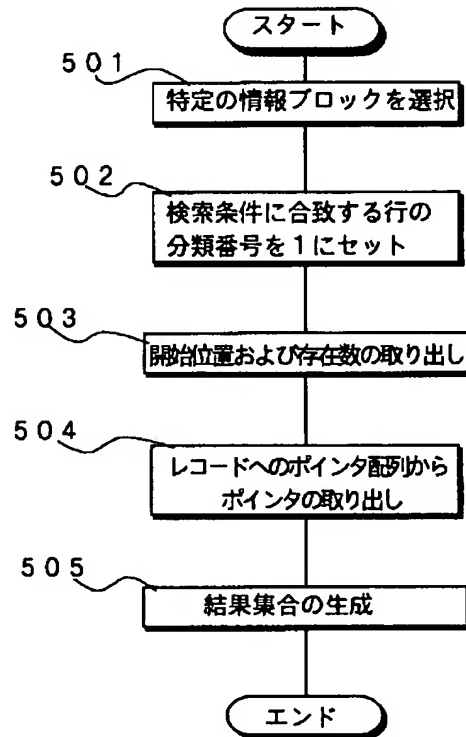
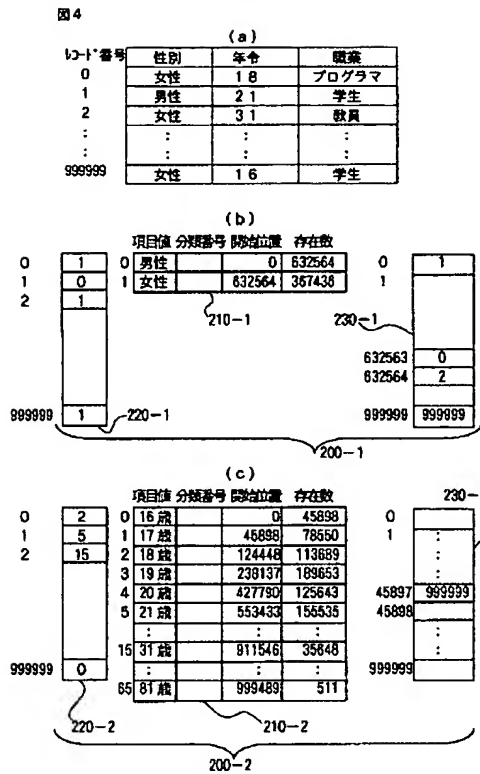


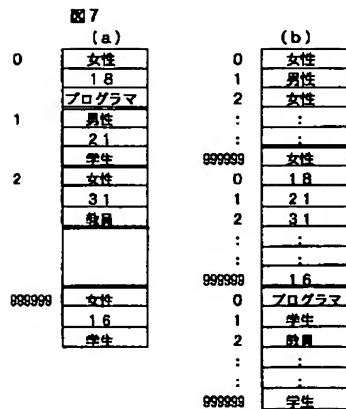
図5



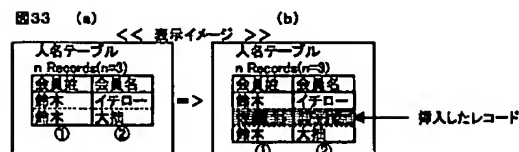
【図4】



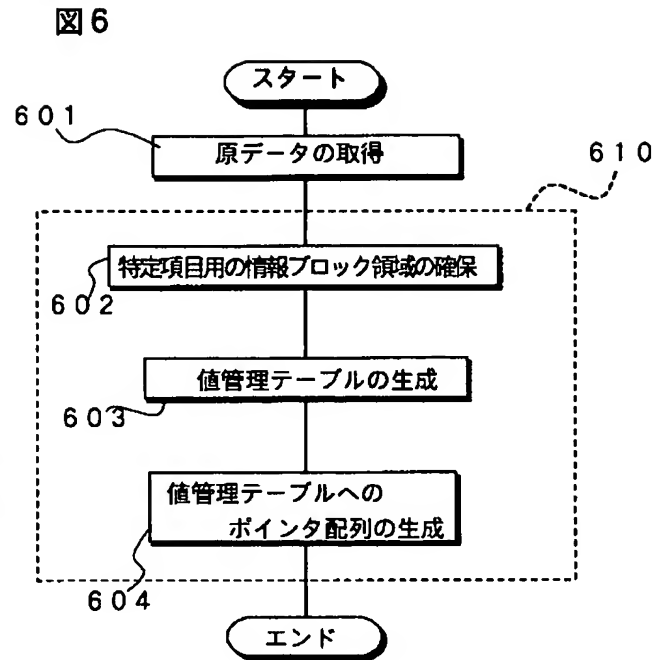
【図7】



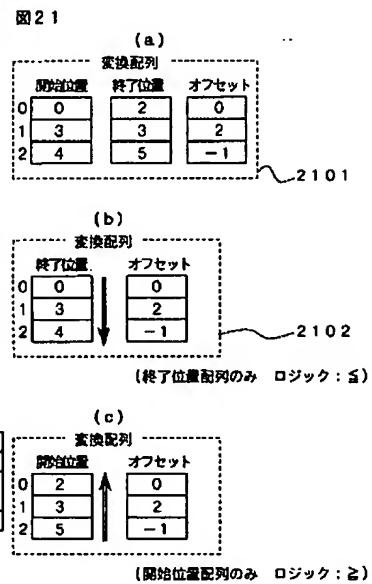
【図33】



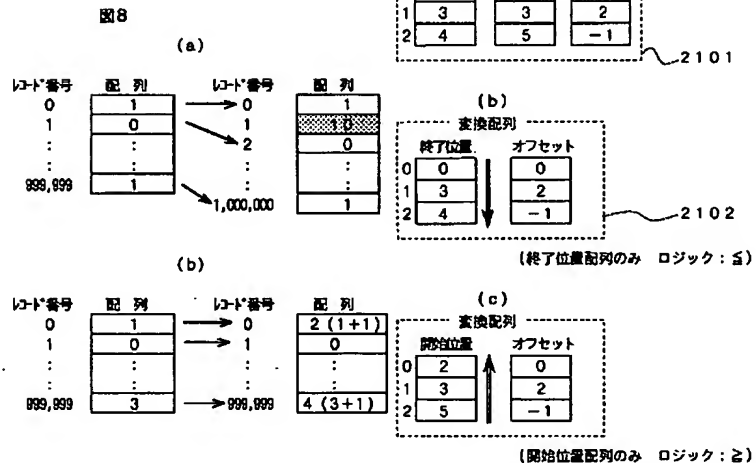
【図6】



【図21】

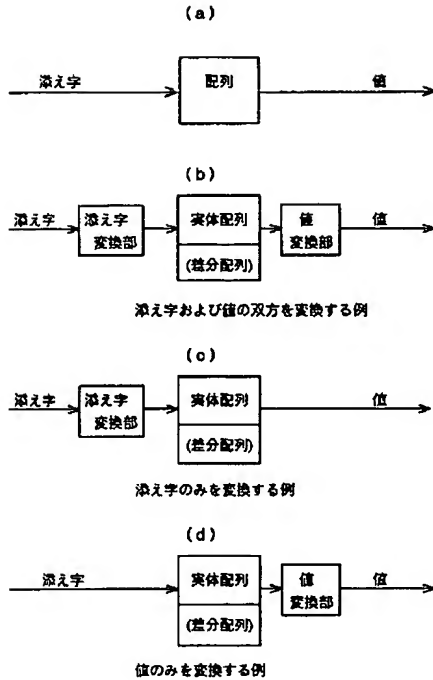


【図8】

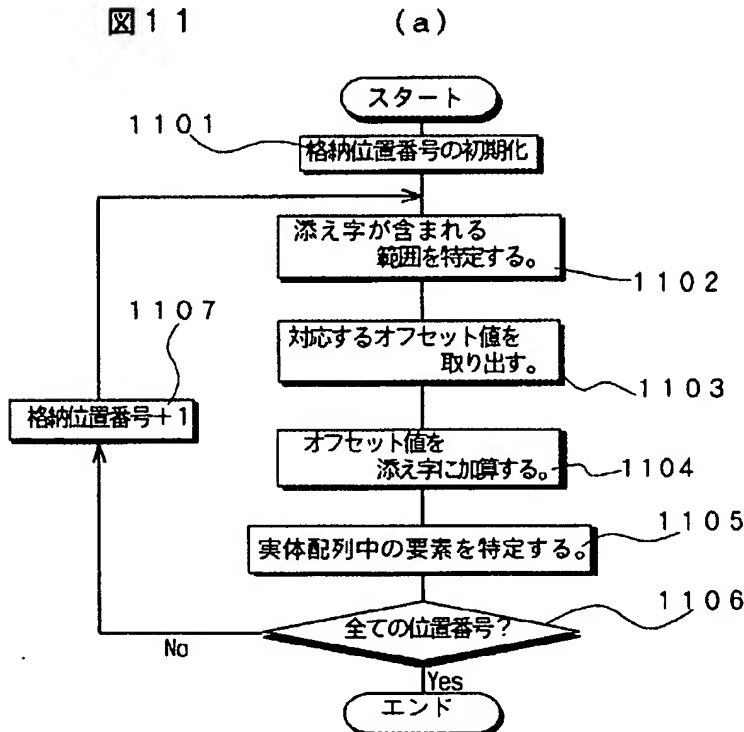


【図9】

図9

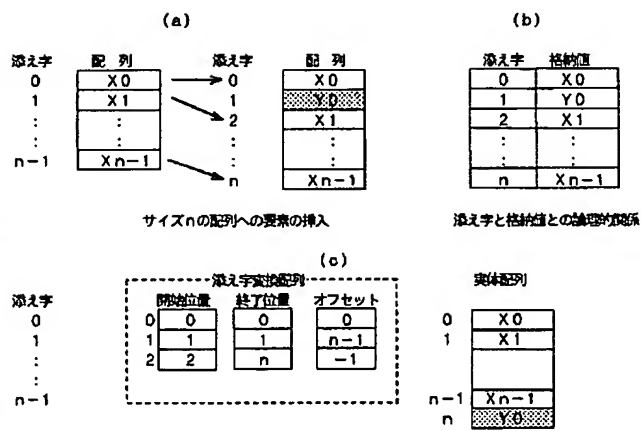


【図11】



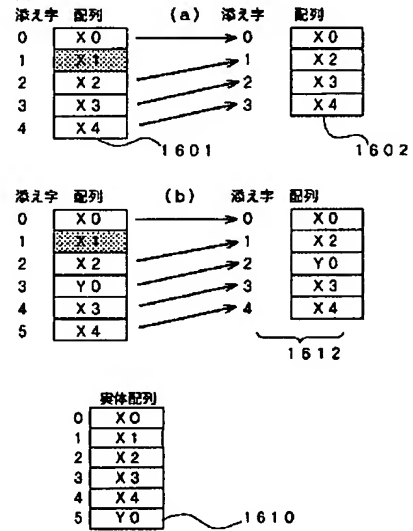
【図10】

図10

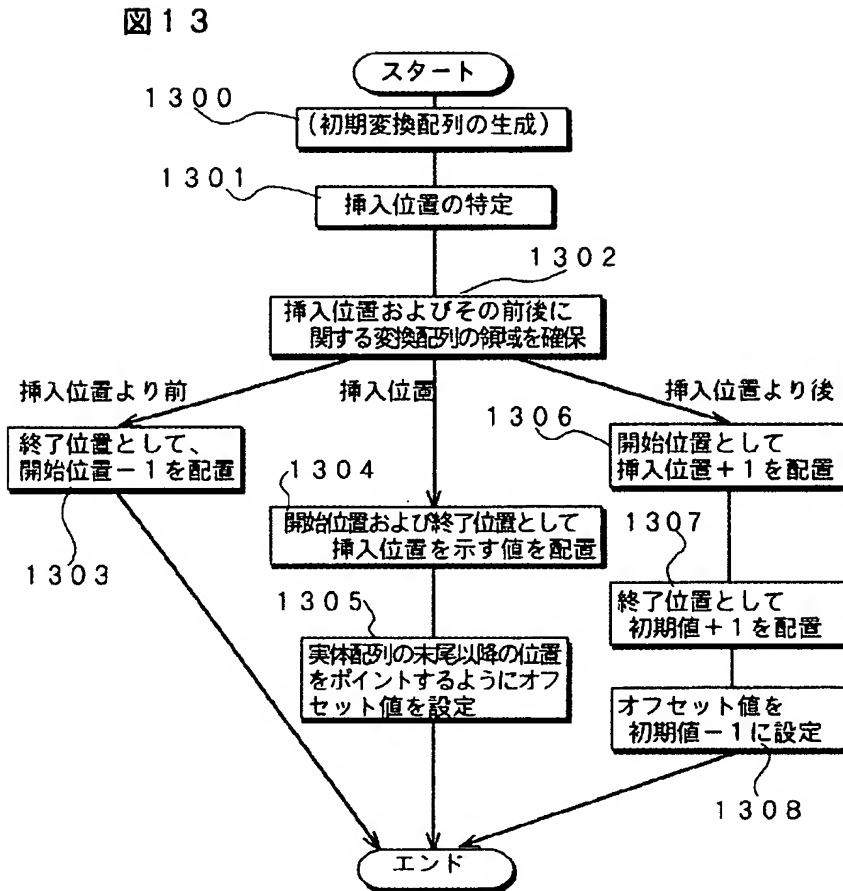


【図16】

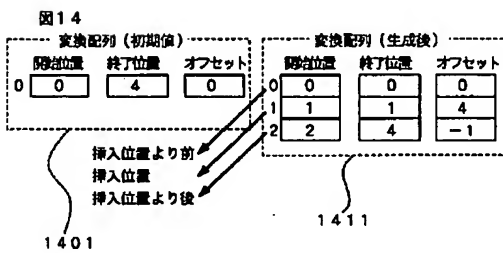
図16



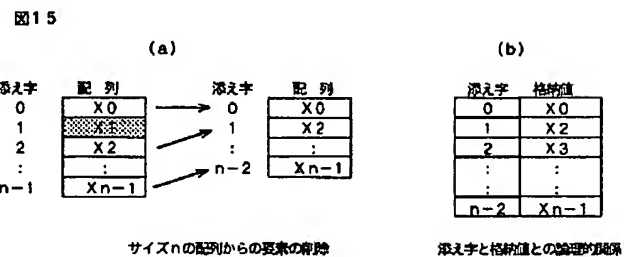
【図13】



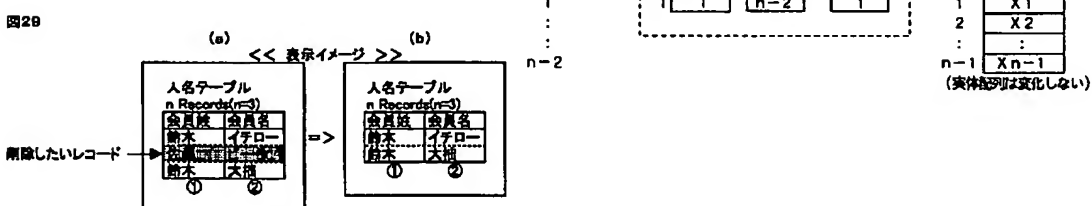
【図14】



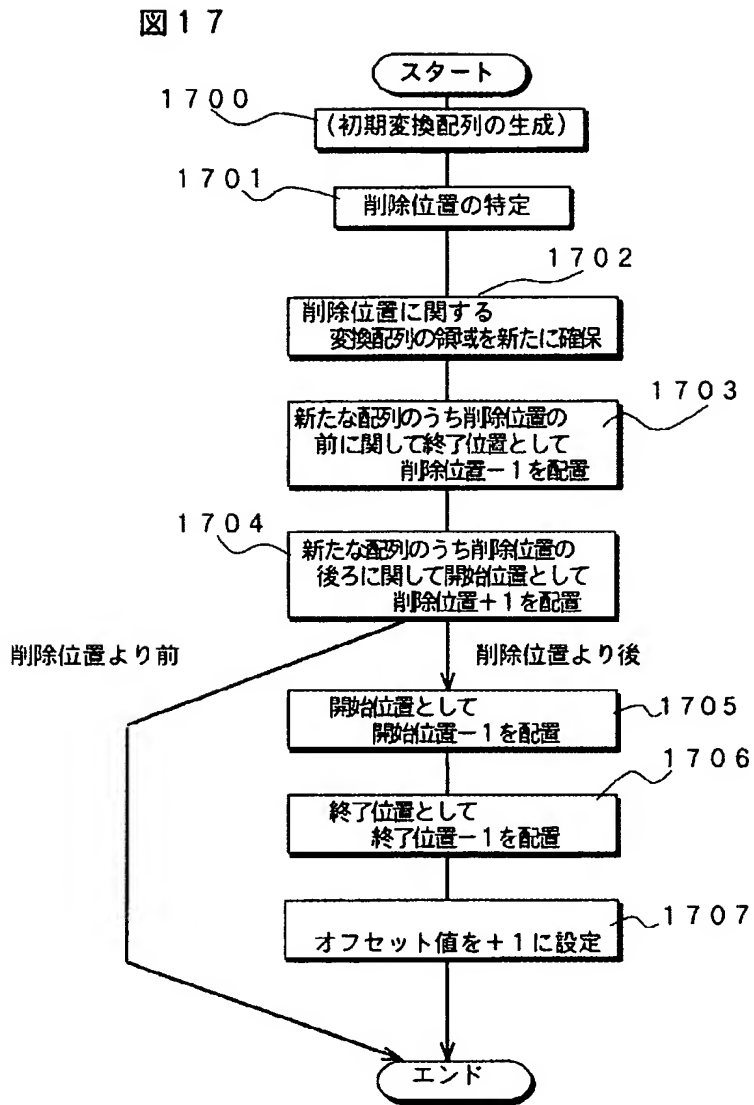
【図15】



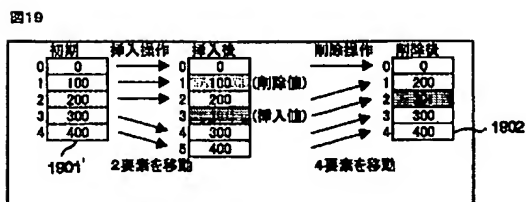
【図29】



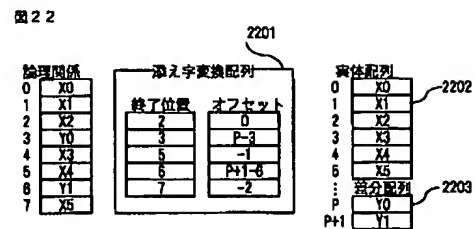
【図17】



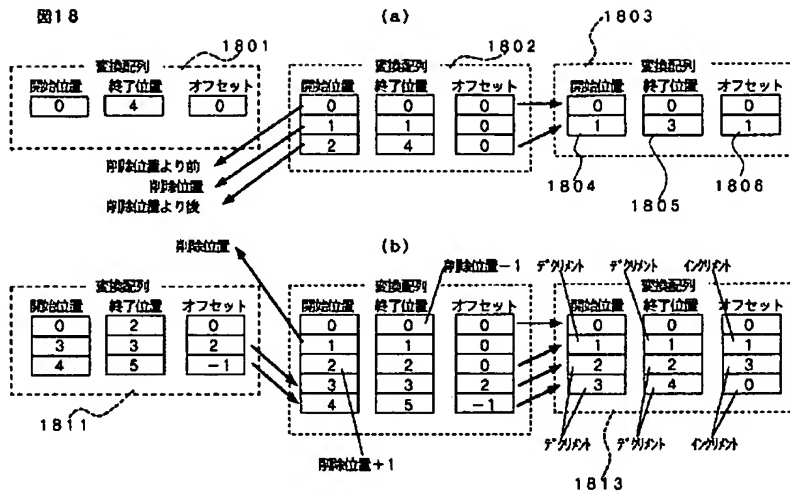
【図19】



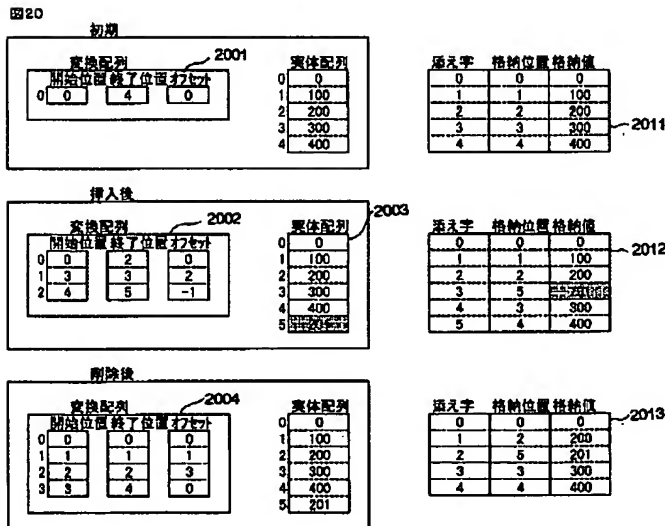
【図22】



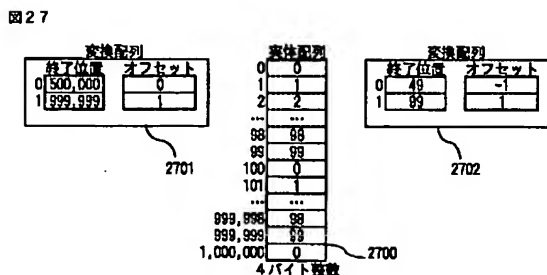
【図18】



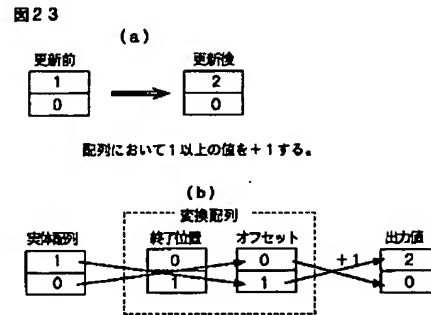
【図20】



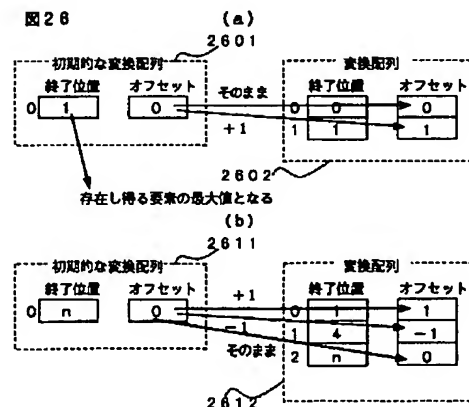
【図27】



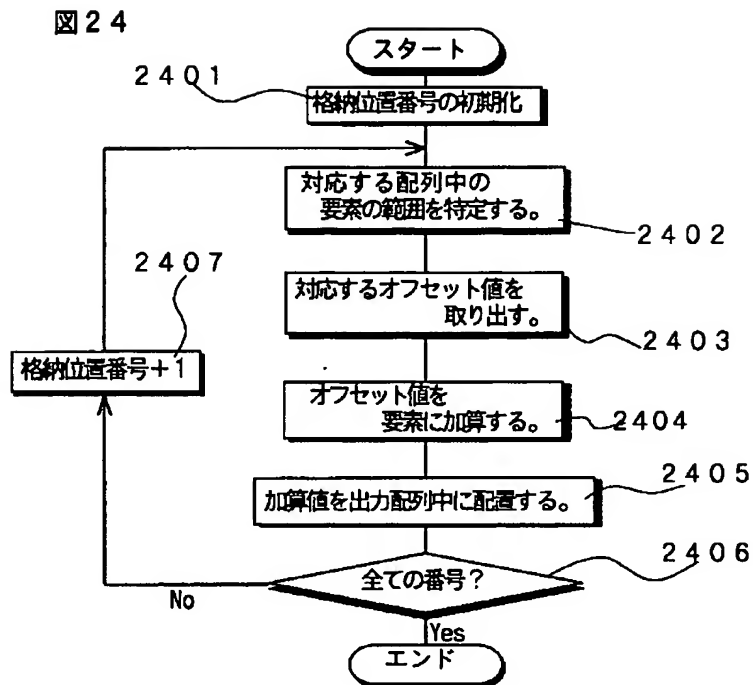
【図23】



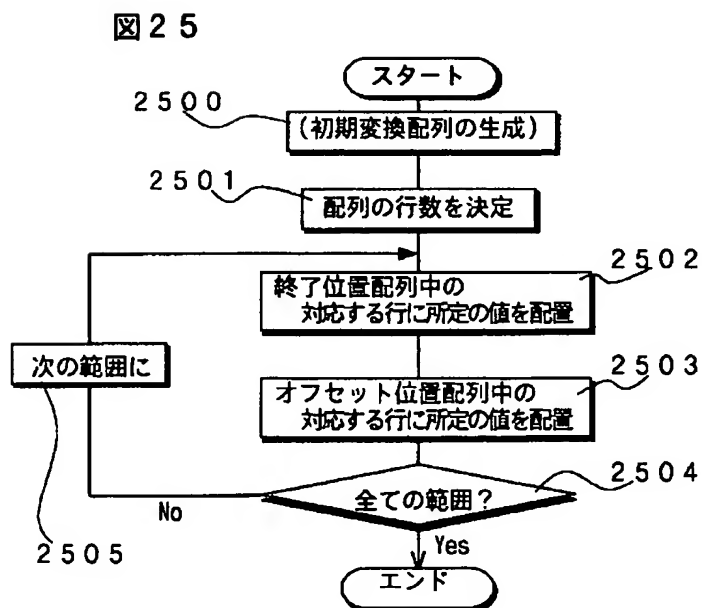
【図26】



【図24】

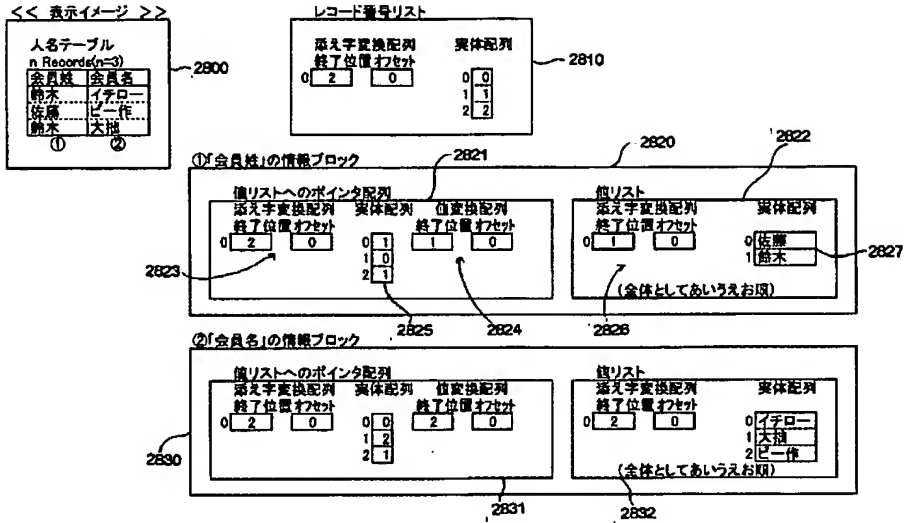


【図25】



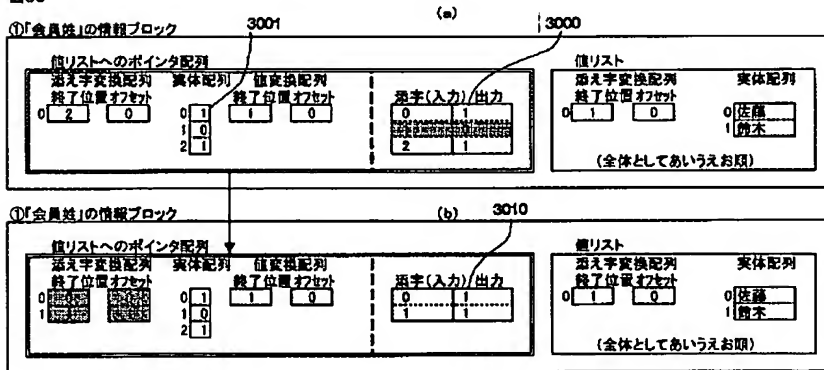
【図28】

図28



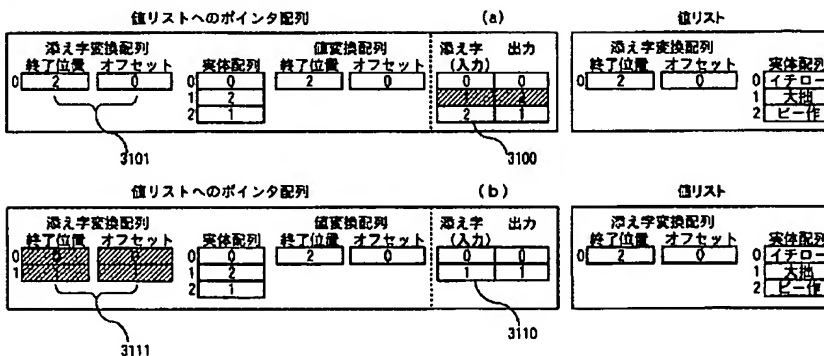
【図30】

図30

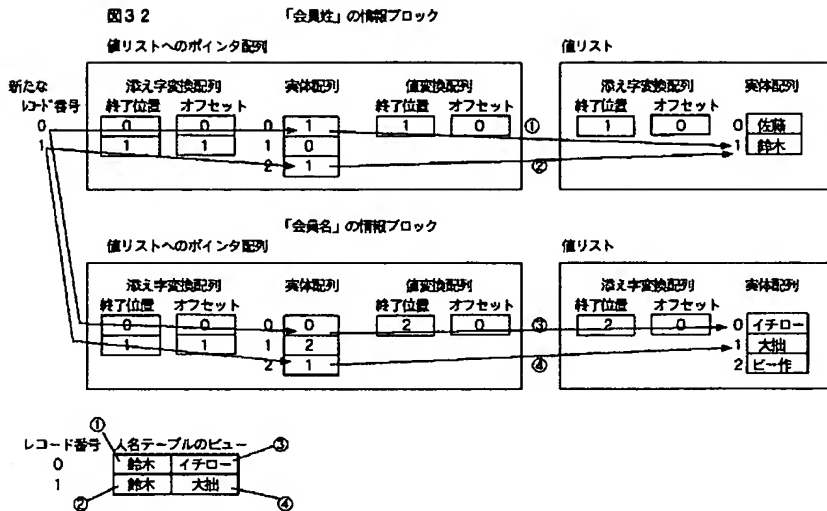


【図31】

図31



【図32】

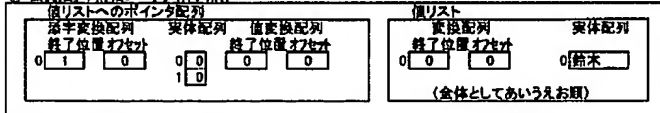


【図34】

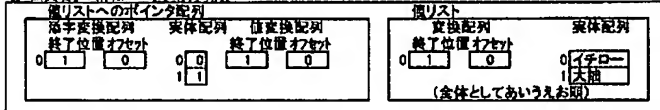
図34
レコード番号リスト(挿入前)

変換配列	終了位置	オフセット	実体配列
0	1	0	0
			1

①「会員姓」の情報ブロック(挿入前)



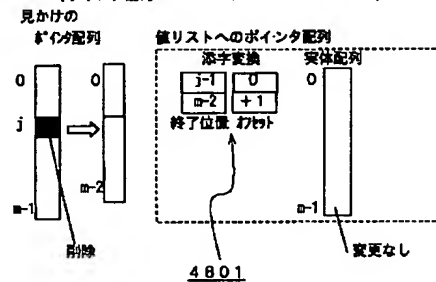
②「会員名」の情報ブロック(挿入前)



【図48】

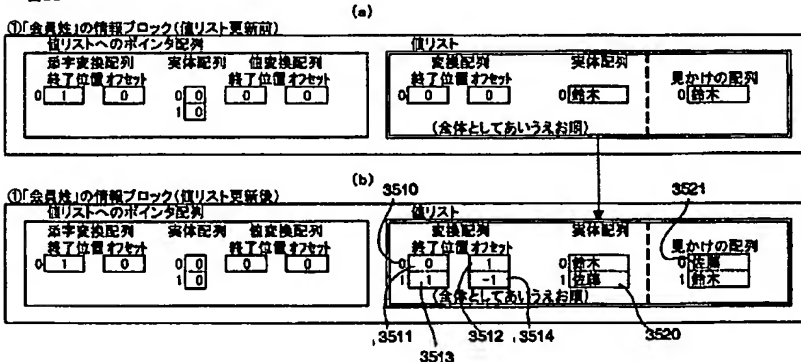
図48

(ポインタ配列: mレコード→m-1レコードへ)

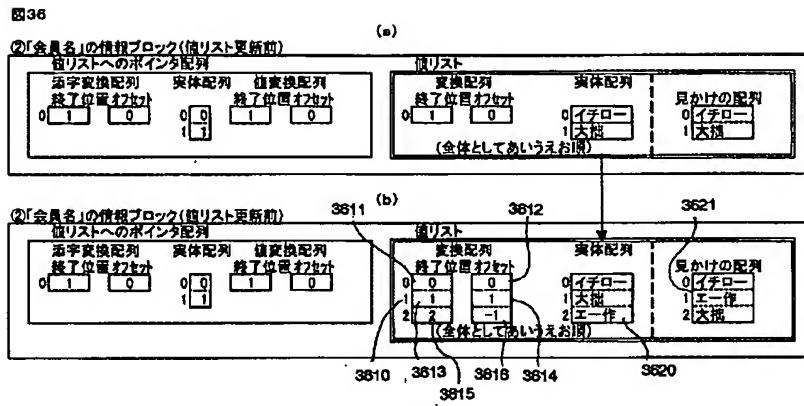


【図35】

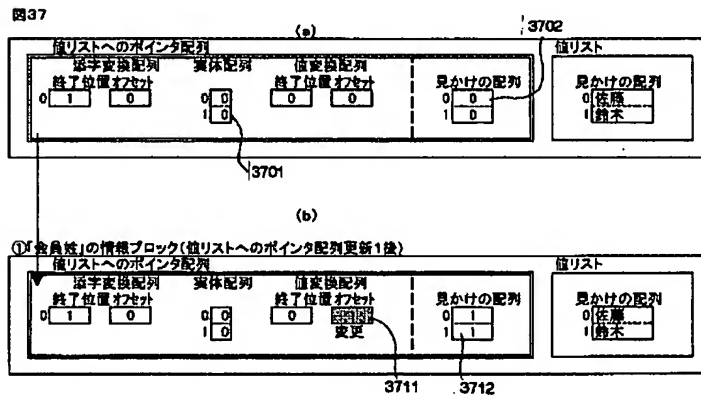
図35



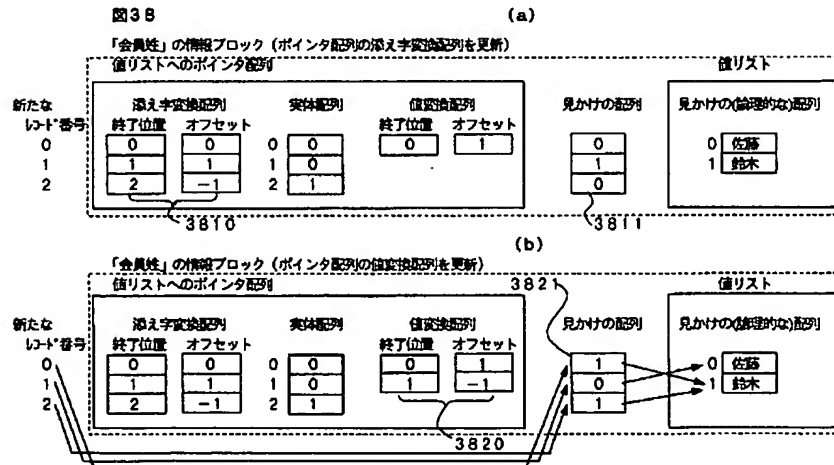
【図36】



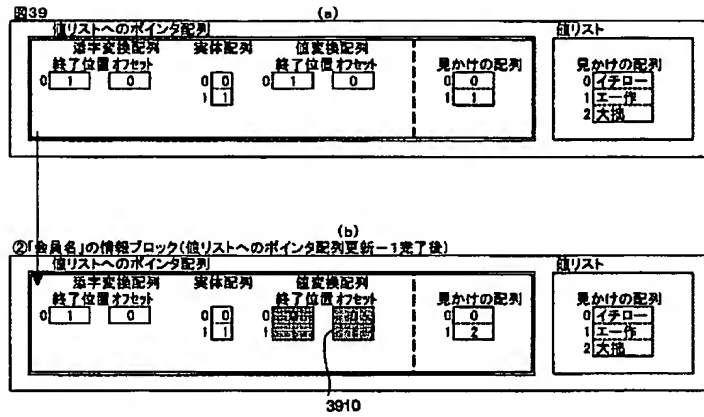
【図37】



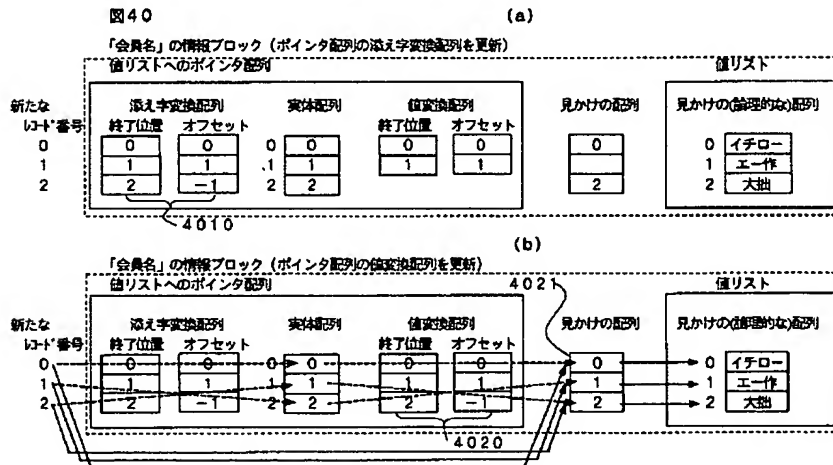
【図38】



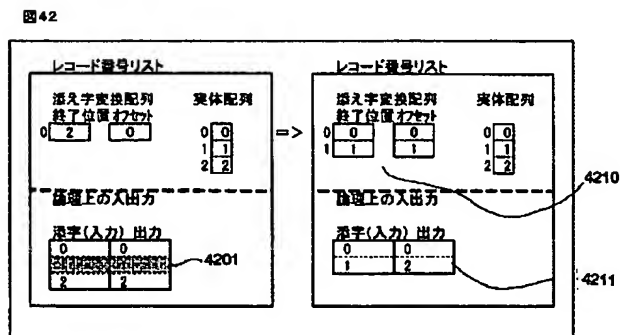
【図39】



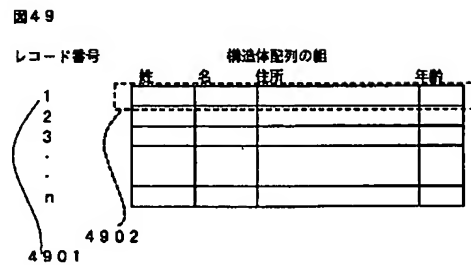
【図40】



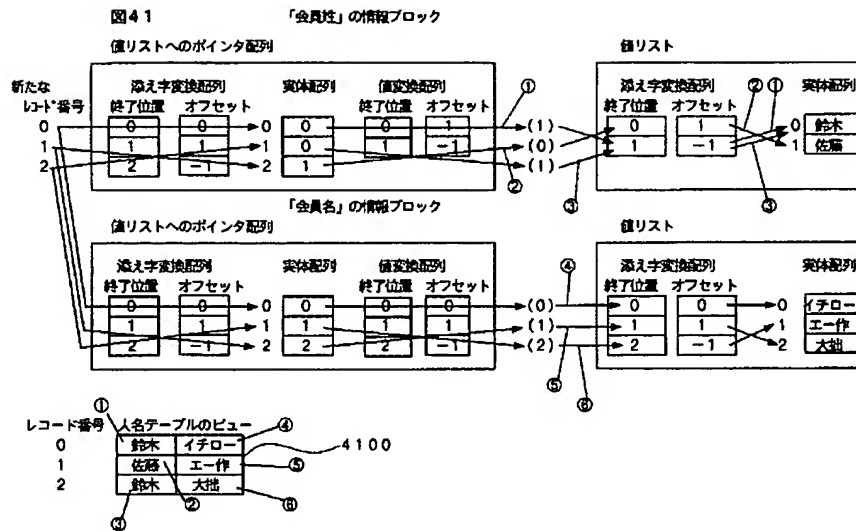
【図42】



【図49】

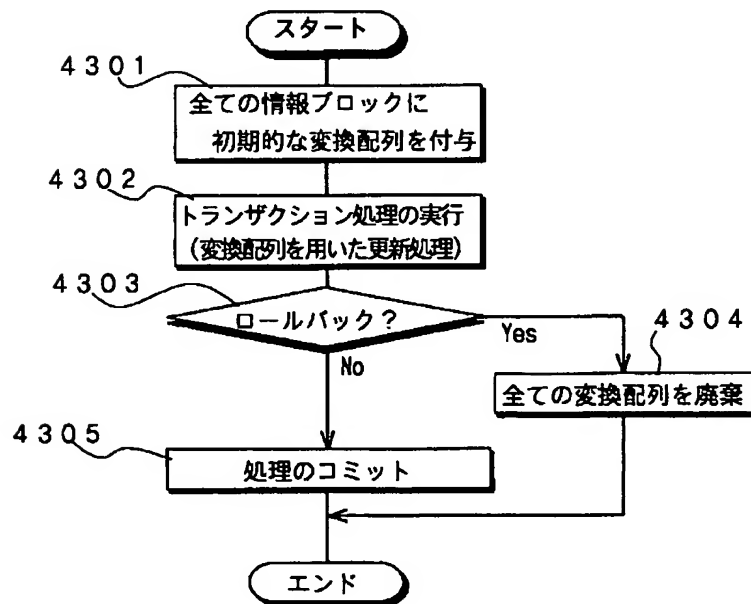


【図41】



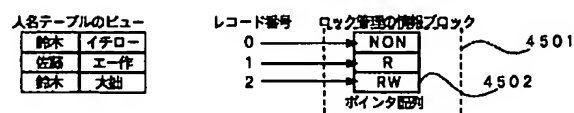
【図43】

図43

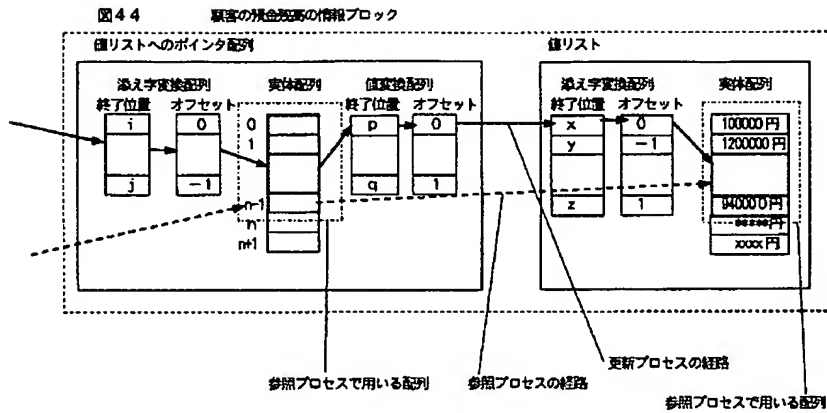


【図45】

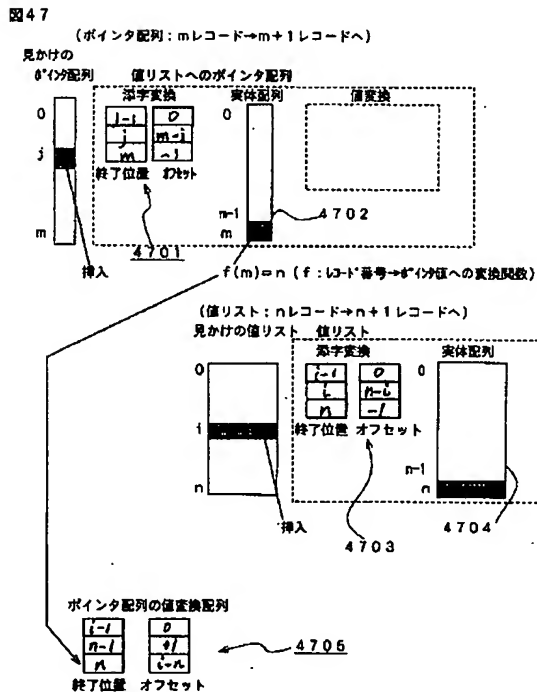
図45



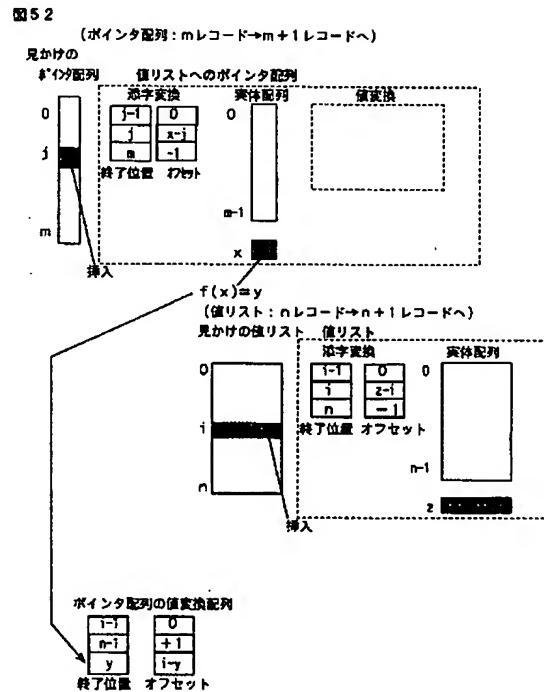
【図44】



【図47】



【図52】



〔図46〕

図46

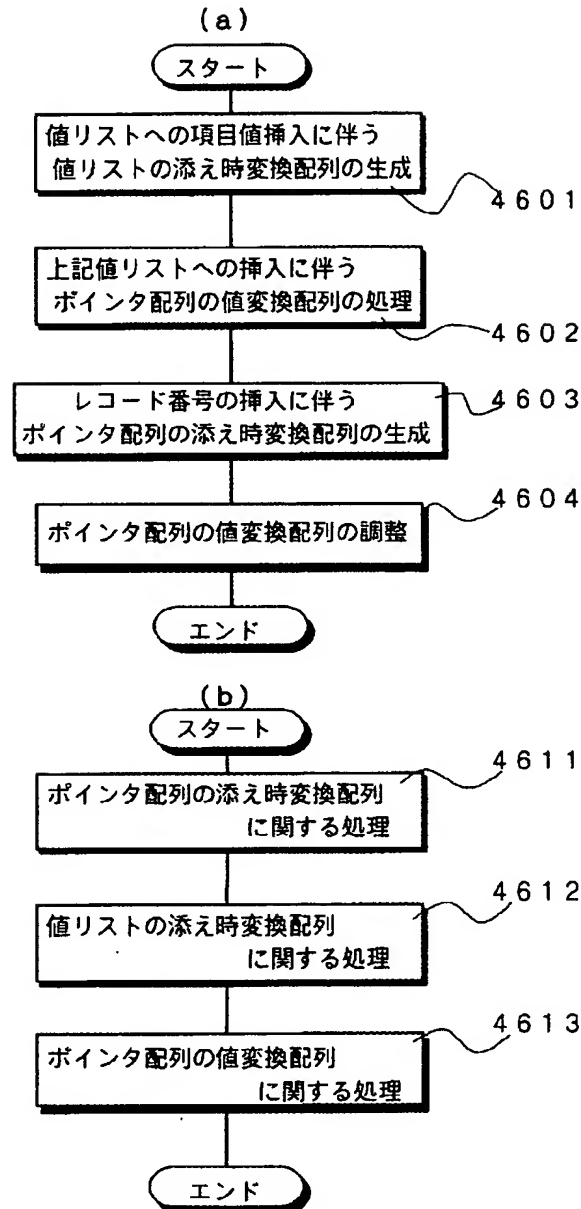
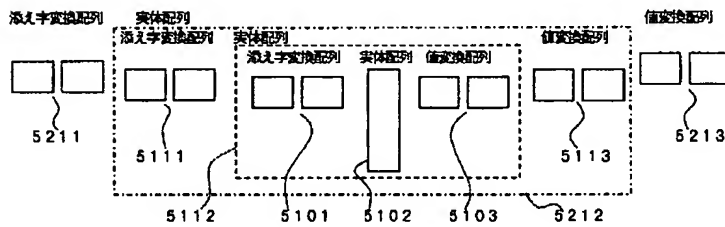


Figure 1 illustrates the data structure of a 100-million-row table. The table is divided into blocks of 10 million rows, labeled 5010, 5011, and 5018. Each block contains a 'Block ID' column and a 'Block Offset' column. The 'Block ID' column contains values from 0 to 99,999,999. The 'Block Offset' column contains values from 0 to 99,999,999. The table is also divided into blocks of 10 million rows, labeled 5010, 5011, and 5018. Each block contains a 'Block ID' column and a 'Block Offset' column. The 'Block ID' column contains values from 0 to 99,999,999. The 'Block Offset' column contains values from 0 to 99,999,999. The table is also divided into blocks of 10 million rows, labeled 5010, 5011, and 5018. Each block contains a 'Block ID' column and a 'Block Offset' column. The 'Block ID' column contains values from 0 to 99,999,999. The 'Block Offset' column contains values from 0 to 99,999,999.

圖 5 1



【図54】

見かけの
ポインタ配列

ポインタ変換

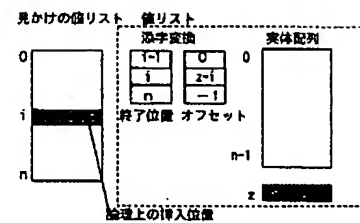
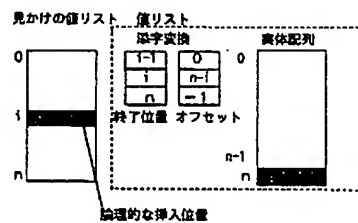
ポインタ配列

値変換

終了位置

更新

$f(m) = n$ (f: コード番号 → ポインタ値への置換関数)



**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☒ FADED TEXT OR DRAWING
- ☒ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.